TESE DE DOUTORADO
UNIVERSIDADE ESTADUAL DE CAMPINAS
FACULDADE DE ENGENHARIA ELÉTRICA E DE COMPUTAÇÃO
DEPARTAMENTO DE ENGENHARIA DE COMPUTAÇÃO E AUTOMAÇÃO INDUSTRIAL

GRAFO DE RELAÇÕES:
UMA METODOLOGIA PARA COORDENAR DEPENDÊNCIAS ENTRE ATIVIDADES EM AMBIENTES COMPUTACIONAIS.

Autor: **Adailton José Alves da Cruz**

Orientador: **Prof. Dr. Léo Pini Magalhães**
Co-orientador: **Prof. Dr. Alberto Barbosa Raposo**

Banca Examinadora:

Prof. Dr. Léo Pini Magalhães – DCA / FEEC / UNICAMP
Prof. Dr. Rafael Santos Mendes – DCA / FEEC / UNICAMP
Prof. Dr. Fernando Antonio Campos Gomide – DCA / FEEC / UNICAMP
Prof. Dr. Ivan Luiz Marques Ricarte – DCA / FEEC / UNICAMP
Prof. Dr. José Carlos Maldonado – ICMC / USP / São Carlos
Prof. Dr. Paulo Eigi Miyagi – PMR / EPUSP / USP / São Paulo

Tese submetida à Faculdade de Engenharia Elétrica e de Computação da Universidade Estadual de Campinas, para preenchimento dos pré-requisitos parciais para obtenção do Título de **Doutor em Engenharia Elétrica**. Área de concentração: **Engenharia de Computação**.

8 de outubro de 2004

FICHA CATALOGRÁFICA ELABORADA PELA
BIBLIOTECA DA ÁREA DE ENGENHARIA - BAE - UNICAMP

Cruz, Adailton José Alves da
C889g Grafo de relações: uma metodologia para coordenar dependências entre atividades em ambientes computacionais / Adailton José Alves da Cruz. -- Campinas, SP: [s.n.], 2004.

Orientadores: Léo Pini Magalhães, Alberto Barbosa Raposo.
Tese (doutorado) - Universidade Estadual de Campinas, Faculdade de Engenharia Elétrica e de Computação.

1. Redes de petri. 2. Grafo (Sistema de computador). I. Magalhães, Léo Pini. II. Raposo, Alberto Barbosa. III. Universidade Estadual de Campinas. Faculdade de Engenharia Elétrica e de Computação. IV. Título.

Resumo

Um dos desafios relacionados à coordenação em ambientes computacionais é a geração de estruturas para equacionar possíveis conflitos decorrentes de relacionamentos de dependências entre as atividades deste ambiente. Este trabalho apresenta uma metodologia para automatizar a geração de mecanismos de coordenação em ambientes computacionais. Estes mecanismos são gerados a partir dos comportamentos temporais especificados para as atividades executadas no ambiente. Permite-se especificar comportamentos temporais alternativos e atividades alternativas, os quais podem ser selecionados em tempo de processamento mudando assim as relações temporais entre as atividades e/ou as atividades que participam destes comportamentos. O algoritmo proposto para este fim implementa uma política de coordenação global permitindo-se que a execução de uma atividade aconteça somente quando não violar qualquer restrição temporal do ambiente. A identificação e modelagem das restrições temporais que cada atividade deve atender resultam no mecanismo de coordenação, obtido em tempo linear no número de atividades. Exploram-se as capacidades de encapsulamento e compactação das redes de Petri coloridas na modelagem dos mecanismos de coordenação sem vincular o uso da metodologia a um conhecimento prévio deste sistema formal (redes de Petri coloridas).

Abstract

One of the challenges related to the coordination of computational environments is the generation of structures to address possible conflicts related to dependences among the activities of these environments. This work presents a methodology to automate the generation of coordination mechanisms for computational environments. These mechanisms are generated from the temporal behaviors specified for the activities executed in the environment. It is possible to specify alternative temporal behaviors and alternative activities, which can be selected in processing time changing temporal relationships among the activities and/or the activities that participate in these behaviors. The algorithm proposed for this implements a global coordination policy, allowing that the execution of an activity only occurs when this doesn't violate any temporal restriction of the environment. The identification and modeling of the temporal restrictions that each activity should satisfy result in the coordination mechanism, obtained in linear time in the number of activities. The modular and compacting capacities of colored Petri nets are explored in the modeling of the coordination mechanisms without connecting the use of the methodology to a previous knowledge of this formal system (colored Petri nets).

Agradecimentos

Ao PICD/UFMS, pelo apoio financeiro através da bolsa de doutorado.

Ao Prof. Dr. Léo Pini Magalhães pela orientação inteligente e presente, ao Prof. Dr. Alberto Barbosa Raposo pela objetividade de suas sugestões e ao Prof. Dr. Rafael Santos Mendes pelos seus valorosos questionamentos.

Ao DCA/FEEC/UNICAMP, pela infra-estrutura.

À minha esposa Maria Aparecida pela torcida no decorrer do trabalho, ao meu filho Henrique por ter ouvido com paciência tantas vezes a frase "o papai tem que trabalhar" como resposta a inúmeros convites como ir ao clube, ir a banca de revista, etc.

A meus pais e irmãos pelo estímulo e por terem sempre compreendido minha ausência em momentos especiais desta caminhada.

A meus tios José e Ana pelas longas e agradáveis conversas degustando um delicioso café de final de tarde.

A todos colegas pelo companheirismo e incentivo buscando saber como estava o trabalho.

A todos os professores que compartilharam suas experiências como pesquisador.

A cidade de Campinas pelos seus bosques, praças, feiras e shoppings.

À minha esposa Maria Aparecida

e aos meus filhos Henrique e Bruna.

ÍNDICE

LISTA DE FIGURAS

LISTA DE DEFINIÇÕES, PROPOSIÇÕES, CRITÉRIOS, LEMAS, TEOREMAS, PROCEDIMENTOS E ALGORITMOS.

1. INTRODUÇÃO

Um ambiente computacional pode ser visto como um conjunto de sistemas formados por processos que, por sua vez são determinados por um conjunto de atividades logicamente relacionadas. As atividades são como as "engrenagens" de uma máquina e, as execuções destas geram os produtos oferecidos pelo ambiente computacional. Por exemplo, a edição de um texto por um grupo de pessoas, uma apresentação multimídia, o resultado de uma pesquisa na web, etc.

As atividades representam fragmentos bem definidos no funcionamento geral da aplicação e guardam relações de dependência entre si. Assim, as execuções destas atividades podem apresentar necessidades e interesses conflitantes que devem ser previstos e atendidos. Por exemplo, um conjunto de atividades deve ser executado simultaneamente ou, o início da execução de uma atividade promove a execução (ou fim da execução) de outras atividades ou ainda, a atividade **a** pode ser executada se e somente se a atividade **b** já foi executada.

Neste contexto, uma das dificuldades é coordenar um conjunto de atividades logicamente relacionadas de forma a atingir os objetivos da aplicação. Esta coordenação pode ser vista como o esforço que deve ser acrescido ao ambiente computacional com o propósito de prever e atender possíveis conflitos derivados dos relacionamentos entre as atividades.

Em [Malone 94] a questão da coordenação é abordada como "o processo de gerenciar dependências entre atividades". Este gerenciamento é realizado sob a forma de mecanismos de coordenação [Crowston 94]. Em outras palavras, a coordenação consiste em gerenciar as dependências definidas entre as atividades estabelecendo mecanismos de coordenação com a função de fazer cumprir os comportamentos fixados por estas dependências.

Por exemplo, em ambientes colaborativos deseja-se organizar a execução das atividades de modo a atingir um objetivo comum, como a redação de um texto por um grupo de pessoas. Em um ambiente multimídia as atividades de reprodução de áudio, vídeo, animação e interação com o usuário devem ser sincronizadas obedecendo às dependências temporais entre elas. Em um ambiente de animação por computador, onde os atores são autônomos, deseja-se que estes atuem no ambiente obedecendo a um conjunto de comportamentos pré-estabelecidos.

Considerando as definições anteriores ([Malone 94] e [Crowston 94]), os mecanismos de coordenação modelam os comportamentos que devem ser coordenados no sistema. Estes comportamentos são descritos através das

dependências entre as atividades. Estas dependências podem ser categorizadas tendo em vista o comportamento que podem expressar, como por exemplo, para as seguintes dependências entre duas atividades *a* e *b*:

categoria	**exemplo de dependência**
temporal	*a* deve ser executada antes de *b*
causal	o início de *a* promove o início de *b*
espacial	*a* é executada se estiver próxima de *b*
produtor-consumidor	*a* produz e *b* consome.

Alguns autores ([Malone 93], [Dellarocas 94]) propõem a construção de um catálogo listando dependências e os seus possíveis mecanismos de coordenação como ferramenta de auxílio ao projetista do sistema.

Um outro ponto refere-se à representação dos mecanismos de coordenação. Alguns autores ([van der Aalst 96], [Raposo 00a], [Zaidi 99], entre outros) usam redes de Petri e justificam citando a quantidade de técnicas disponíveis para simulação, análise, verificação e validação dos mecanismos de coordenação e a sua capacidade de representar tanto formalmente quanto graficamente estes mecanismos. Em [Zaidi 99] discute-se também a questão da consistência da representação, i.e. se a representação pode ser realizada de forma consistente ou se as dependências modeladas são contraditórias. Outras questões são as possíveis dificuldades impostas pelo sistema formal ao projetista do sistema. Por exemplo, a manipulação deste formalismo pode ser trabalhosa se não existir uma semântica intuitiva para a especificação do sistema computacional [Wahl 94].

No presente trabalho estuda-se e propõe-se uma nova metodologia, denominada Grafo de Relações ou metodologia **GR**, para tratar a questão da coordenação. Dentre os objetivos deseja-se alcançar:

- um processo para automatizar a modelagem do mecanismo de coordenação;
- mecanismos de coordenação livres de inconsistências temporais;
- uma representação formal dos mecanismos de coordenação que não vincule o uso da metodologia a um conhecimento prévio do sistema formal.

Para o processo de geração dos mecanismos de coordenação propõe-se um algoritmo, de complexidade linear *O(n)*, sendo *n* o número de atividades, para automatizar a identificação e modelagem das restrições temporais. Estas restrições são derivadas dos relacionamentos de dependência especificados entre as atividades. O resultado é um mecanismo de coordenação com as seguintes características:

- a execução de qualquer atividade nunca viola nenhuma restrição temporal;

- seleção de comportamento - possibilidade de selecionar diferentes comportamentos (dependência entre atividades) para um mesmo subconjunto de atividades;
- seleção de atividades – possibilidade de selecionar se um subconjunto de atividades será ou não executado.

O modelo de coordenação proposto pela metodologia **GR** é baseado em níveis de abstração: nível de especificação, nível de coordenação e nível de execução. No nível de especificação descrevem-se os comportamentos temporais entre as atividades usando um conjunto de primitivas adotadas de [Allen 84] e conceitos de teoria dos grafos.

No nível da coordenação procede-se a identificação e modelagem das restrições temporais obtendo-se o mecanismo de coordenação **MC**. O **MC** é modelado por uma Rede de Petri, gerada a partir da descrição dos comportamentos temporais.

No nível da execução cada atividade deve ser associada ao código correspondente a sua função. Os aspectos de implementação do nível de execução não são tratados no decorrer do texto, pois se entende que para coordenação importa apenas a identificação e modelagem das relações entre as atividades e não como estas devem ser implementadas.

Abordar a questão da coordenação inserindo um nível de abstração entre a descrição da aplicação e sua execução possibilita:

- usar processos de descrição dos comportamentos temporais independentes do sistema formal empregado para modelar os mecanismos de coordenação;
- determinar quando e qual atividade será executada (coordenação do trabalho) sem entrar em detalhes de modelagem da atividade (como o trabalho é executado).

A Metodologia **GR** não pressupõe uma aplicação específica. O objetivo é propor uma metodologia para obtenção de mecanismos de coordenação considerando os conceitos de atividade, dependência e mecanismo de coordenação. O mecanismo destina-se a dar suporte à coordenação das dependências temporais estabelecidas entre as atividades da aplicação.

O capítulo 2 apresenta uma discussão sobre coordenação em ambientes computacionais sob a óptica de algumas aplicações de interesse enfocando contribuições de autores e técnicas correlatas a este trabalho.

O capítulo 3 apresenta a metodologia **GR** em detalhe e demonstra sua capacidade de gerar automaticamente um Mecanismo de Coordenação Global, i.e. as atividades são executadas se forem satisfeitas as restrições temporais impostas diretamente e indiretamente a elas.

O capítulo 4 apresenta exemplos de uso dos conceitos desenvolvidos no capítulo 3 através de estudos de caso.

Por fim, o capítulo 5 finaliza o trabalho apresentando as conclusões a respeito da metodologia **GR** e sugestões para trabalhos futuros.

2. COORDENAÇÃO EM DIFERENTES ÁREAS DE APLICAÇÃO

A necessidade de tratar a questão da coordenação de atividades de um ambiente computacional surge das dependências (temporais, de recurso, mútua exclusão, etc.) existentes entre estas atividades. Estas dependências colocam restrições e condições para a execução das atividades e assim, é necessário imprimir um esforço adicional, o trabalho da coordenação, para gerenciar tais dependências a fim de prever-se e atender-se possíveis conflitos no ambiente.

Este tema tem sido alvo de diversas investigações científicas ([Malone 94], [Schimidt 96], [Raposo 00a],[Crowston 00] etc) e vem despertando o interesse dos pesquisadores na busca por ferramentas que auxiliem a coordenação de um ambiente computacional.

Este capítulo apresenta trabalhos relacionados à coordenação de ambientes computacionais e que servirão de base ao presente estudo. Dar-se-á ênfase àqueles que fazem uso de redes de Petri para sua modelagem. Para fins de apresentação o enfoque será por áreas de aplicação.

Na seção seguinte adota-se uma definição para coordenação e apresentam-se alguns conceitos gerais a respeito do tema. As demais seções caracterizam e discutem a necessidade da coordenação em alguns campos de pesquisa apresentando-se trabalhos de maior relevância para o contexto da metodologia apresentada no Capítulo 3.

2.1. Conceitos Gerais

Em ambientes computacionais a execução das atividades pode apresentar necessidades e interesses conflitantes que devem ser previstos e atendidos. Nesta perspectiva, a questão colocada é como *coordenar* as atividades de forma a atingir os objetivos do ambiente.

A palavra *coordenação* significa "estabelecer relações entre elementos de forma que funcionem de modo articulado dentro de uma totalidade ordenada" [Ferreira 99]. Esta definição geral fornece uma noção intuitiva do que é coordenação, i.e. existe a necessidade de coordenação quando se observa que a ação das várias partes que compõem um ambiente, sem uma coordenação, não permite atingir as finalidades previstas.

Algumas situações do cotidiano ilustram esta noção intuitiva, por exemplo, a conta de telefone deve ser entregue antes do seu vencimento para que o pagamento possa ser efetuado sem multas. Se isto acontece tem-se um exemplo de coordenação,

um atraso na entrega revela uma falta de coordenação. Outros exemplos são: o ônibus que faz o seu itinerário no horário previsto, o som harmônico proveniente da regência de uma orquestra, um restaurante onde não se espera demasiadamente para ser atendido, o fluxo de carros em uma malha rodoviária, entre outras.

Uma noção mais precisa de coordenação é apresentada em **[Malone 94]**. Ele trata a questão da coordenação como sendo "o processo de gerenciar dependências entre atividades". Isto é consistente, pois se não existem dependências entre as atividades não existe a necessidade do trabalho de coordenação.

Sob a ótica da coordenação, as *atividades* representam as partes funcionais da aplicação, as dependências descrevem as relações entre as atividades e os *processos de coordenação* representam o protocolo de interconexão que gerenciam os relacionamentos e restrições derivados das dependências [Malone 94].

Assim, um ambiente computacional pode ser visto como sistemas de atividades interdependentes gerenciadas por processos de coordenação. Para ilustrar esta afirmação, a Figura 2-1 apresenta a arquitetura de software de uma aplicação de processamento de documentos baseada no sistema TeX [Dellarocas 96]. As caixas representam as atividades e as linhas que as conectam as respectivas dependências. O processo de coordenação é o mecanismo que garante o cumprimento das dependências (temporal, de recurso, mútua exclusão, etc) entre as atividades.

Neste sentido, a coordenação de um ambiente computacional requer dois passos principais: primeiro identificar os tipos de dependências entre as atividades e segundo determinar processos de coordenação para gerenciar estas dependências, i.e. mecanismos para prever e atender os possíveis conflitos gerados pelas dependências.

Para auxiliar as tarefas identificar/determinar foi proposto em [Malone 93] e [Dellarocas 94] a construção de um repositório, i.e. um "catálogo", relacionando processos de coordenação obtidos de diferentes áreas do conhecimento (Ciência da Computação, Pesquisa Operacional, Psicologia, Economia, entre outras) e os prováveis tipos de dependências que poderiam ser tratados por eles. Por exemplo, um processo de coordenação usado pela Ciência da Computação para gerenciar dependências entre atividades que competem por um recurso limitado é o mecanismo "primeiro a chegar - primeiro a ser atendido". Este "catálogo" tem uma conotação genérica e não discute que formalismo usar para representar os processos de coordenação.

A construção de um banco de padrões dependência-processo de coordenação justifica-se pelo fato das dependências transcenderem o limite de uma aplicação, i.e. um mesmo tipo de dependência pode acontecer em diferentes aplicações. Por exemplo, o compartilhamento de recursos é uma dependência que pode aparecer em uma aplicação multimídia ou em um ambiente virtual colaborativo e pode ser gerenciada usando-se o mesmo mecanismo "primeiro a chegar - primeiro a ser

atendido". O mesmo não se aplica às atividades, pois para a coordenação cabe gerenciar as dependências entre elas sem envolver-se com os detalhes de sua modelagem.

Um ponto de grande importância para coordenação é a representação dos mecanismos de coordenação. A este tema estão ligadas questões como o tipo de formalismo que deve ser usado para representá-los; se esta representação é consistente; etc.

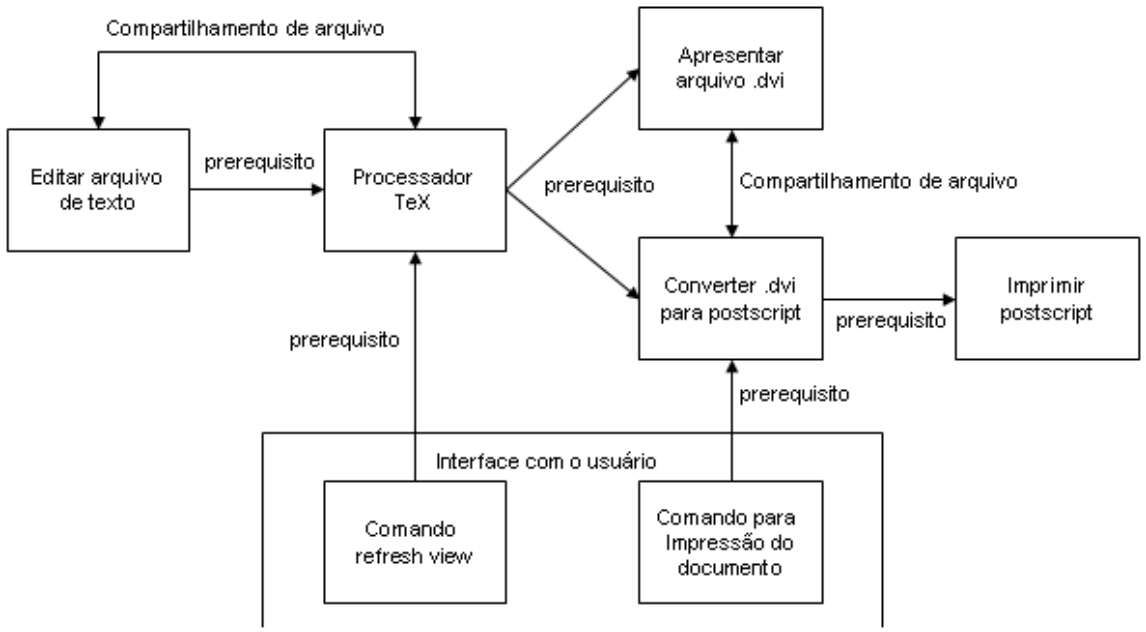


Figura 2 - 1: Aplicação representada através das dependências entre atividades.

Nas próximas seções apresentam-se, de forma geral e segundo a definição de coordenação introduzida aqui (atividade–dependência–mecanismo de coordenação), alguns campos de pesquisa e problemas abordados por eles. Tem-se por objetivo relacionar alguns trabalhos de relevância para o presente, discutir a necessidade da coordenação e dessa forma motivar as contribuições que serão apresentadas no capítulo 3.

2.2. A questão da coordenação em Multimídia

Multimídia é uma nomenclatura genérica para referenciar aplicações de computador que permitem a integração de informações, tais como texto, vídeo,

áudio, animação e imagens. Estas aplicações podem estar organizadas por conexões, sendo, neste caso, denominadas hipermídia [Maurer 93], [Villanova 00].

Um dos problemas a equacionar neste tipo de aplicação é a sincronização (temporal, espacial, entre outras) entre as atividades (reprodução de áudio, vídeo, som, entre outras) da apresentação multimídia. A sincronização é o processo responsável por estabelecer o ordenamento das atividades obedecendo aos tipos de dependências especificadas entre elas [Courtiat 95], [Walh 94]. Esta definição caracteriza a sincronização como um processo de coordenação das dependências entre as atividades da apresentação multimídia.

As dependências potencialmente envolvidas entre as atividades da aplicação devem ser gerenciadas por mecanismos de coordenação que garantam a correta representação e cumprimento das restrições oriundas destas dependências e assim atender às condições de sincronização.

Vários modelos para sincronização dos comportamentos temporais de aplicações multimídia têm sido propostos [Senac 95], [Yoon 98], [Zaidi 99], [Hsu 03], entre outros. Estes modelos empregam diferentes ferramentas matemáticas na sua construção, como por exemplo, modelos baseados em redes de Petri, em grafos, em lógicas temporais, etc., bem como a combinação destas técnicas.

Apresentam-se a seguir, alguns modelos que fazem uso das redes de Petri como sistema formal na representação dos mecanismos de coordenação ilustrando-se pontos de interesse tais como:

- a capacidade deste sistema de encapsular os detalhes que não são de interesse da coordenação, como por exemplo, os detalhes de modelagem das atividades;
- o emprego de dependências temporais para modelar os processos de sincronização;
- as possibilidades de interação com o usuário da aplicação.

Em [Senac 95] propõe-se um sistema denominado "Hierarchical Time Stream Petri Net" – HTSPN. As atividades da aplicação são representadas pelos lugares da rede, os arcos especificam as restrições temporais aplicadas às atividades. A capacidade de encapsular das redes de Petri é explorada em três níveis de abstração. O primeiro nível destina-se à especificação formal do sistema hipermídia. As atividades mais complexas ("Composite Component") são detalhadas em uma sub-rede em um segundo nível de modelagem. Quando uma atividade atinge o nível atômico ("Atomic Component"), por exemplo a reprodução de um vídeo, as informações pertinentes a ela são descritas em um terceiro nível de abstração. A Figura 2 - 2 adaptada de [Senac 95] ilustra o processo de modelagem descrito acima.

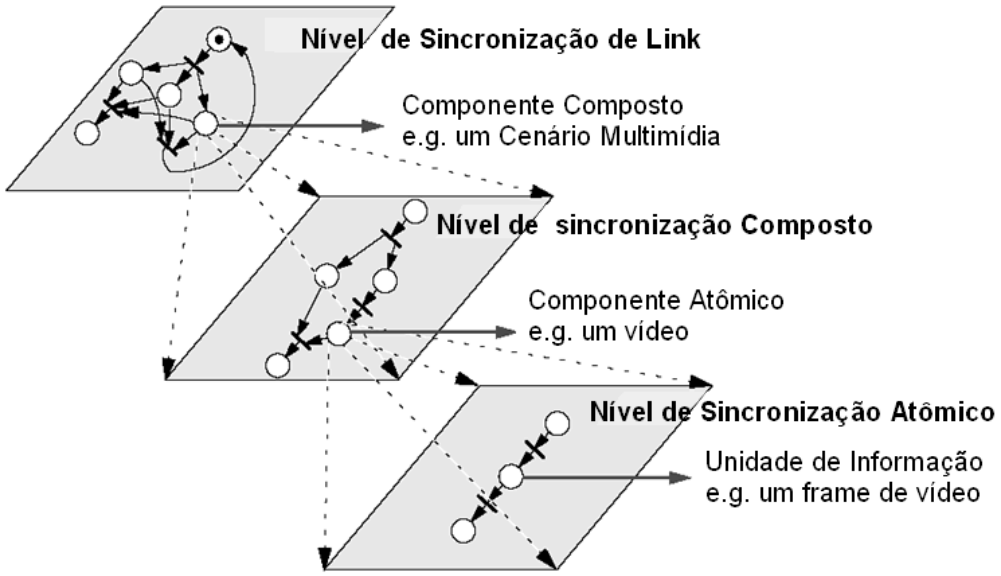


Figura 2 - 2 : Níveis de sincronização do modelo apresentado por [Senac 95].

Quanto ao modelo apresentado em [Senac 95]:

a) uma grande vantagem da modelagem por meio de níveis de abstração é o seu caráter modular. Isto oferece maior facilidade na troca ou atualização do mecanismo de sincronização de um componente composto, pois esta operação é localizada e tende a envolver apenas o elemento a ser atualizado.

b) a representação da aplicação através de um sistema formal como as redes de Petri, possibilita ao projetista da aplicação explorar processos de análise e verificação para correção e validação da aplicação efetuando-se simulações dos módulos de interesse. Estas simulações permitem uma redução considerável dos erros, eventualmente inseridos em um sistema mais complexo, durante a especificação dos comportamentos temporais. A relação sistema formal e simulação não é uma questão trivial [Cassandras 93]. Este fato pode ser a motivação para o desenvolvimento de alternativas para tratar tal questão

c) as redes de Petri são especificadas diretamente pelo projetista e não existe um processo de geração automática para elas. Conseqüentemente isto requer mais tempo do projetista, existe a possibilidade de inserir erros, é cansativo e vincula o uso da metodologia a um conhecimento prévio dos conceitos do sistema formal empregado.

O sistema apresentado em [Hsu 03] tem como foco a coordenação de objetos multimídia móveis ("moving multimedia objects") como o exemplo dado a seguir. Considere uma figura **B** que deve manter-se abaixo e a dois centímetros do vértice inferior esquerdo de uma figura **A**, sendo que **A** se desloca segundo uma dada curva. Uma das dificuldades neste tipo de aplicação é a ausência de estruturas eficientes capazes de: 1) modelar as restrições temporais e os atributos espaciais (distância entre objetos, tamanho, posição relativa, entre outros) e 2) viabilizar o fluxo destes atributos entre os objetos relacionados. O trabalho em questão usa uma extensão das redes de Petri, denominada STRPN ("Spatial and Temporal Relationship Petri Nets"), na modelagem de estruturas de transferência de atributos espaciais. A Figura 2 - 3 ilustra uma aplicação multimídia modelada em STRPN.

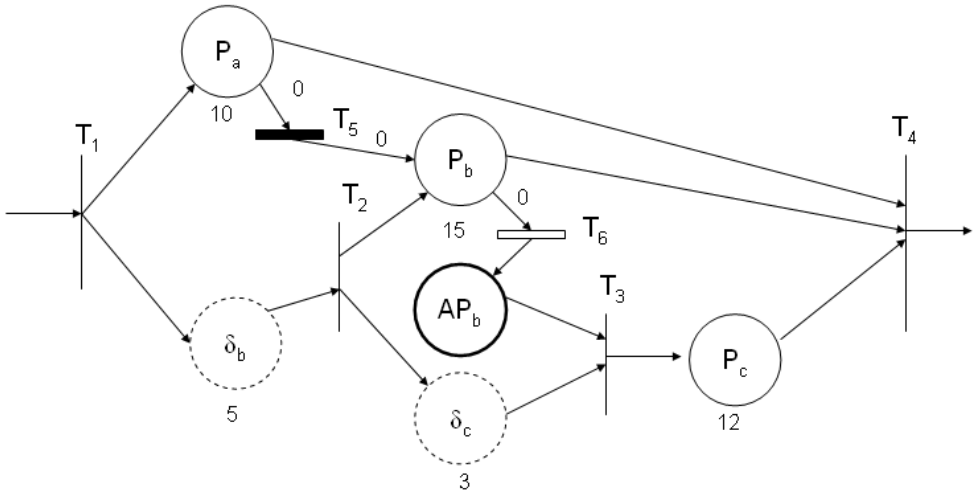


Figura 2 - 3: Exemplo de uma aplicação multimídia modelada em STRPN.

O Diagrama da Figura 2 - 3 ilustra as três classes de lugares: lugares que representam objetos multimídia (P_a, P_b e P_c), lugares para armazenar informações espaciais (AP_b) e lugares de retardo (δ_b e δ_c). As transições também são classificadas em três categorias: as transições usadas para transmitir informações espaciais entre objetos relacionados (dois tipos - T_5 e T_6) e as transições que não transmitem informações espaciais (de T_1 a T_4). As regras e condições de disparo são definidas para cada classe de transição.

Um aspecto interessante do processo de modelagem através das STRPN é a integração de restrições temporais e restrições espaciais em um mesmo mecanismo de coordenação.

O TOCPN ("Transitional Object Composition Petri Net") é o modelo apresentado em [Yoon 98] para especificação dos comportamentos temporais e para o processo de sincronização de aplicações multimídia. A formalização deste modelo se dá por redes de Petri onde tanto os lugares, que representam as atividades, como os arcos são temporizados. As dependências entre as atividades são descritas atribuindo-se valores temporais a estes elementos. Uma atividade pode ter tempo de execução indeterminado. Neste caso o fim de execução é especificado em função das outra(s) atividade(s). Por exemplo, se duas atividades *a* e *b* são executadas em paralelo, sendo que uma delas *a* tem tempo de execução indeterminado *u*, então o fim da execução desta atividade *a* pode ser determinado em função do tempo de execução *d* da outra atividade *b*, como ilustra a Figura 2-4.

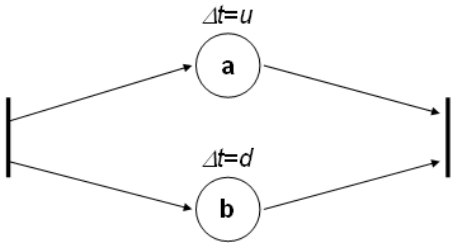
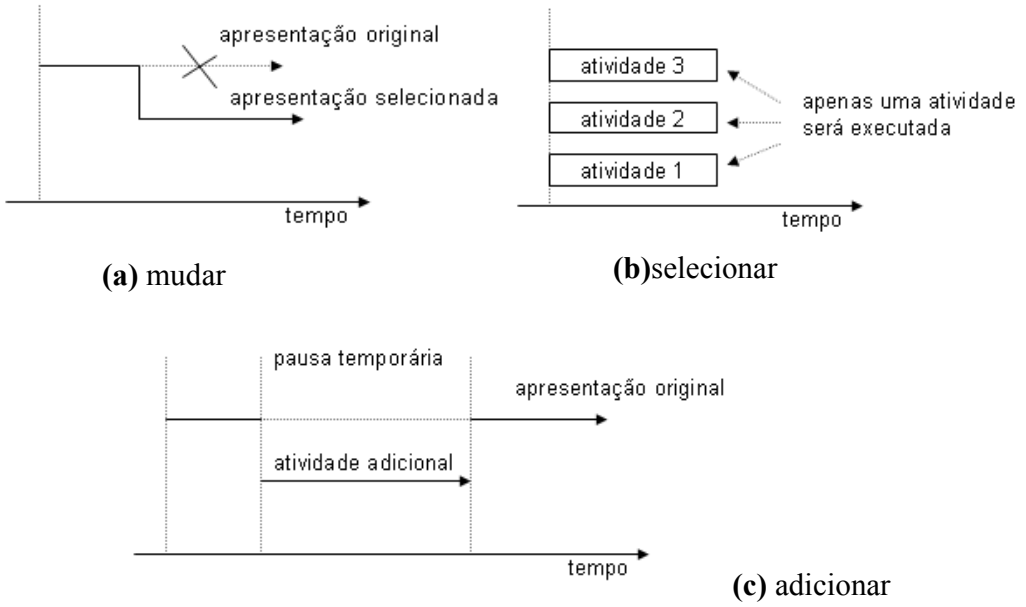


Figura 2-4: Tempo de execução de *a* depende do tempo de execução de *b*.

O foco principal do trabalho de Yoon é o conjunto de possibilidades de interação entre o usuário e a aplicação. São oferecidas três classes de interação que permitem:

1. Mudar o curso da apresentação do documento multimídia, encerrando-o e iniciando a apresentação de um outro documento (Figura 2-5(a));
2. Selecionar uma atividade dentre um conjunto de atividades possíveis (Figura 2-5(b));
3. Inserir uma atividade adicional, devidamente especificada, à apresentação do documento multimídia(Figura 2-5(c)).

Evidencia-se aqui a possibilidade de associar um intervalo de tempo não determinado a uma atividade. Esta capacidade confere ao modelo maior poder de expressão permitindo modelar situações desejáveis em aplicações multimídia. Como exemplo, considere-se a situação em que uma atividade é executada até que o usuário solicite seu fim, neste caso o intervalo de tempo associado a ela deve ter duração não determinada. Esta capacidade permite explorar uma classe de eventos importantes na modelagem da aplicação, aqueles cujo instante de ocorrência não é conhecido a priori.



(a) mudar **(b)**selecionar

(c) adicionar

Figura 2 - 5: Formas de interação

Em [Zaidi 99] desenvolveu-se uma metodologia para a especificação de dependências temporais e inferências de novas relações temporais entre as atividades de uma aplicação. A metodologia introduz um modelo conceitual que é uma extensão da lógica temporal apresentada em [Allen 83]. A aplicação é descrita usando o modelo conceitual que é transformado em uma Rede de Petri. Para verificar se a especificação é consistente e inferir novas relações obtém-se da Rede de Petri um grafo direcionado. A especificação é dita consistente se o grafo não contém ciclos[1].

Embora seja permitido especificar intervalos de tempo com duração não nula, estes são traduzidos para pares de pontos (instantes de tempo). Todos estes pontos são alinhados sobre um único eixo do tempo, estabelecendo uma ordem (<, > ou =) entre eles. Esta característica obriga o modelo a conhecer a priori todos os eventos da aplicação. Esta falta de flexibilidade implica em menor poder de expressão. Por exemplo, um evento determinado interativamente não pode ser expresso, pois não se conhece a priori o instante de tempo em que irá acontecer [Wahl 94]. O modelo também determina que todas as atividades especificadas devem ser executadas, isto é, não é possível expressar comportamentos com seqüências alternativas.

[1] " A descrição de um sistema contém informações inconsistentes se dados dois pontos (instantes de tempo) t_1 e t_2 (representando início e/ou fim dos intervalos de tempo) as expressões seguintes são verdadeiras: $t_1 < t_2$ e $t_1 = t_2$ ou $t_1 < t_2$ e $t_1 > t_2$ ", estas expressões podem levar a ocorrência de ciclos no grafo [Zaidi 99].

O ponto de destaque do trabalho apresentado em [Zaidi 99] é o sistema para inferir novas dependências temporais a partir das conhecidas. Inferir novas dependências ou determinar, para um conjunto de atividades e as dependências entre elas, se existe uma dependência redundante não é trivial [Allen 83]. No entanto, sabendo-se responder estas a questões, pode-se simplificar processos de coordenação ou resolver problemas de inconsistências da seguinte forma: identificar possíveis dependências redundantes e verificar se a remoção destas resolvem os problemas citados.

2.3. A coordenação em CSCW

Dentre as definições encontradas na literatura para o acrônimo inglês CSCW - Computer Supported Cooperative Work, uma das mais freqüentes é a apresentada em [Bannon 91]: "CSCW deve ser percebido como um esforço para entender a natureza e as características do trabalho cooperativo com o objetivo de projetar tecnologias adequadas baseadas em computador". Em outras palavras, CSCW tem por objetivo:

- estudar como acontece o trabalho em grupo[2];
- prover tecnologias que favoreçam a interação e a colaboração entre os participantes de um grupo de trabalho.

O estudo do trabalho em grupo envolve diferentes áreas do conhecimento, tais como psicologia, sociologia, engenharia de software, ciência da computação e antropologia na busca do entendimento, conceitualização e formalização a respeito do tema. Para uma discussão mais extensa a propósito do tema sugere-se [Schmidt 91], [Bannon 94] e [Cartensen 96].

As tecnologias de hardware e software resultantes destes estudos são denominadas em inglês "groupware". O exemplo mais conhecido de groupware é o correio eletrônico que possibilita uma interação assíncrona distribuída (tempos diferentes e lugares diferentes) entre os participantes. Outro exemplo é a aplicação de videoconferência que permite uma interação síncrona distribuída (mesmo tempo e lugares diferentes) entre seus participantes. Sistemas baseados na tecnologia "Groupware (também denominada tecnologia de colaboração) são sistemas nos quais a tecnologia de comunicação e computação apóia grupos de participantes e ajuda a realizar um ambiente compartilhado" [Ellis 99]

Um dos aspectos do CSCW é a coordenação do ambiente compartilhado na realização de um trabalho. A complexidade na realização de um trabalho pode exigir

[2] "o trabalho cooperativo ocorre quando múltiplos atores são exigidos para a realização de um trabalho e portanto tornam-se mutuamente dependentes nos seus trabalhos e devem coordenar e integrar suas atividades individuais para que o trabalho seja realizado." [Shimidt 91].

recursos que estão além das capacidades de um ator[3], necessitando assim da colaboração de outros para que o mesmo seja efetuado. Esta necessidade gera relações de dependência entre as atividades realizadas por estes atores [Crowston 94]. Por sua vez, as dependências devem ser gerenciadas de modo a eliminar ou evitar possíveis conflitos entre atividades.

Assim, a coordenação caracteriza-se por equacionar as restrições impostas pelas dependências provendo um conjunto de funcionalidades[4], denominado coordenador [Ellis 99] ou mecanismo de coordenação [Schmidt 96], com a finalidade de administrar a evolução temporal do sistema determinando quando uma atividade está habilitada[5].

Em CSCW os primeiros mecanismos de coordenação foram construídos a partir de protocolos muito rígidos com componentes fixadas de forma a não permitir modificação ou troca. Como conseqüência tiveram um emprego limitado [Schmidt 96]. A partir destas experiências os pesquisadores investiram em modelos de coordenação que contemplassem a flexibilidade na realização do trabalho. Por exemplo, é desejável que o modelo permita que os atores possam estabelecer o tempo de execução das atividades, escolher quem executa, alterar a seqüência em que as atividades são executadas ou mudar a relação de dependência entre elas. Como em sistemas de apoio a reuniões virtuais, embora exista uma seqüência de atividades definidas a priori, os participantes podem estabelecer quem vai executar qual atividade e qual o tempo para efetuá-las [Ellis 99].

Apresenta-se nos próximos parágrafos o Coordinator, uma aplicação CSCW que faz parte desta geração de aplicações cooperativas com protocolos mais rígidos e finaliza-se apresentando um exemplo de mecanismo de coordenação usado pelo Coordinator.

Coordinator

O Coordinator é uma aplicação cooperativa que apóia a comunicação entre atores [Winograd 86]. Um ator tem conhecimento sobre os outros atores e pode estabelecer uma conversa com qualquer outro. No entanto esta conversação segue uma dinâmica pré-estruturada e que não pode ser alterada.

O sistema pressupõe uma equivalência entre linguagem e ação admitindo que a conversação possa ser construída a partir de um conjunto limitado de tipos de ação pré-definidas. Por exemplo, uma mensagem de A para B deve ter um cabeçalho padrão especificando o tipo de ação (uma requisição, uma rejeição, entre outros tipos de ação) adicionado de um texto com informações que o usuário julgar

[3] Um ator pode ser um usuário, um sistema de computador ou um grupo.
[4] Descrição dos relacionamentos entre as atividades, do tempo de execução, quando uma atividade será executada e por qual ator, entre outras funcionalidades.
[5] Uma atividade está habilitada, i.e. pode ser executada, quando todas as dependências relacionadas a ela tiverem sido satisfeitas. Por exemplo, um banco só deve efetuar um crédito para um cliente depois de executar outras atividades relacionadas.

necessárias. A troca de mensagem é estabelecida segundo um protocolo, representado por um autômato finito, como o exemplo ilustrado na Figura 2-6. A partir desta rede básica formam-se as redes de troca de mensagens. Uma mensagem de A para B começa por uma *requisição*. B tem como alternativa responder com uma *promessa* de cumprir o solicitado, uma *rejeição* ao que lhe foi requisitado ou com uma *contra proposta*. Assim, a rede de conversação vai sendo definida com base nas solicitações e respostas.

Neste tipo de abordagem supõe-se que o processo de colaboração entre os participantes de um grupo de trabalho segue um padrão. Ou seja, a comunicação entre dois elementos do grupo, durante a realização de uma atividade, é uma seqüência de "atos da fala" que inicia com uma requisição que pode ser respondida com uma promessa e assim por diante [Ljungberg 95].

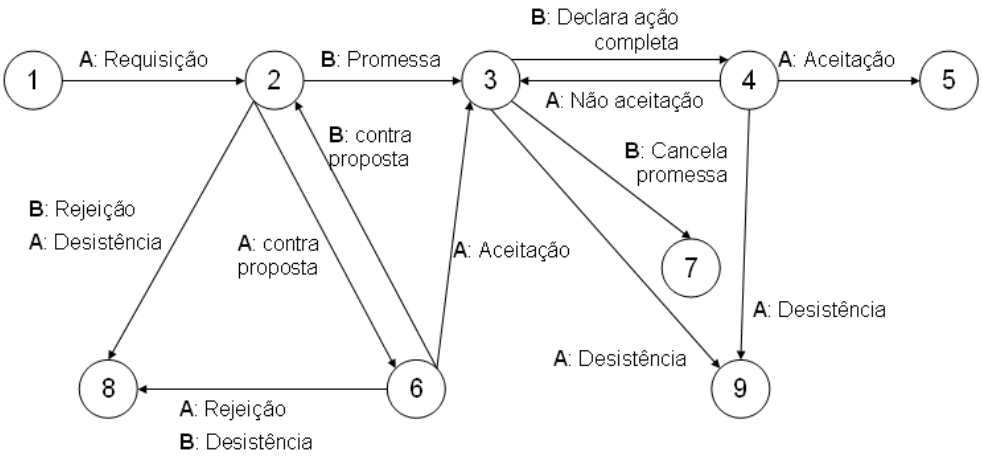


Figura 2-6: Mecanismo de Coordenação do Coordinator.

Do ponto de vista da coordenação A comunica-se com B e B comunica-se com A; A e B relacionam-se através das dependências (requisição, promessa, entre outras) indicadas na Figura 2-6 e as restrições derivadas por estas dependências são modeladas por um autômato finito (mecanismo de coordenação). Este mecanismo estabelece um mapeamento um-para-um em uma conversa, e isto pode limitar a interação entre os participantes de um grupo de trabalho gerando um distanciamento entre o que se pretende e o que é realizado. Dessa forma, buscam-se relações de dependência com capacidade de mapeamento um-para-vários que podem oferecer maior interação entre os participantes de um ambiente cooperativo.

Mecanismos de Coordenação em ambientes colaborativos usando redes de Petri

Em [Raposo 00a] apresenta-se um conjunto de mecanismos de coordenação para gerenciar dependências temporais e dependências de recurso entre os participantes de um ambiente colaborativo. Para cada participante projeta-se uma Rede de Petri modelando suas atividades e as dependências entre elas. Na seqüência, definem-se as possíveis dependências entre participantes diferentes, concluindo-se o primeiro nível de abstração, denominado pelo autor de *nível de workflow*. Considere um exemplo onde dois participantes (usuário A e usuário B) interagem por meio de um "whiteboard[6]" compartilhado. Os usuários devem requisitar simultaneamente o uso do whiteboard - dependência de recurso entre as atividades entraA e entraB (Simultaneidade 2); apenas um usuário pode escrever de cada vez - dependência de recurso entre as atividades escrevaA e escreveB (Exclusão Mútua); e por fim a dependência temporal entre as atividades saiA e saiB estabelece que o canal de comunicação deve ser fechado assim que o usuário A finalizar sua atividade (Meets). A Figura 2 - 7 ilustra o nível de workflow para este exemplo.

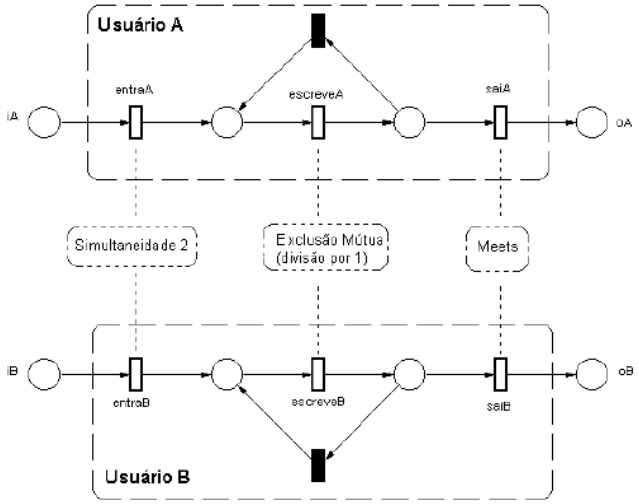


Figura 2 - 7 : Whiteboard – Nível de workflow.

Uma vez projetado o nível de workflow o modelo do sistema é obtido automaticamente. Isto é, em um segundo nível de abstração, denominado *nível de coordenação*, as atividades, representadas pelas transições, são expandidas segundo um modelo pré-definido. Os mecanismos de coordenação, também pré-definidos, correspondentes às dependências especificadas são inseridos, obtendo-se o modelo

[6] Um Whiteboard é um recurso geralmente presente em aplicações colaborativas que possibilita a interação em tempo real entre usuários. Um whiteboard pode ser um espaço no monitor de cada usuário remoto onde um ou mais usuários podem escrever ou desenhar usando dispositivos de entrada como o mouse ou teclado, por exemplo.

do sistema colaborativo. A Figura 2-8 apresenta o modelo obtido para o exemplo do whiteboard compartilhado.

A separação entre atividades e dependências e o uso de mecanismos de coordenação pré-definidos tem como uma das vantagens a geração automática do modelo do sistema a partir do nível de workflow. Assim, o projetista preocupa-se apenas em especificar o sistema no nível de workflow. As vantagens de uma abordagem em níveis de abstrações e o uso de redes de Petri na representação do modelo do sistema são as mesmas já apresentadas na Seção 2.2 itens a) e b).

Os mecanismos de coordenação apresentados para as dependências temporais permitem relacionar apenas duas atividades e, portanto garantem apenas o cumprimento das restrições temporais derivadas por esta dependência. Uma análise global e a verificação de inconsistências temporais (comportamentos temporais que não podem ser realizados) devem ser feitas pelo projetista do sistema no nível de workflow.

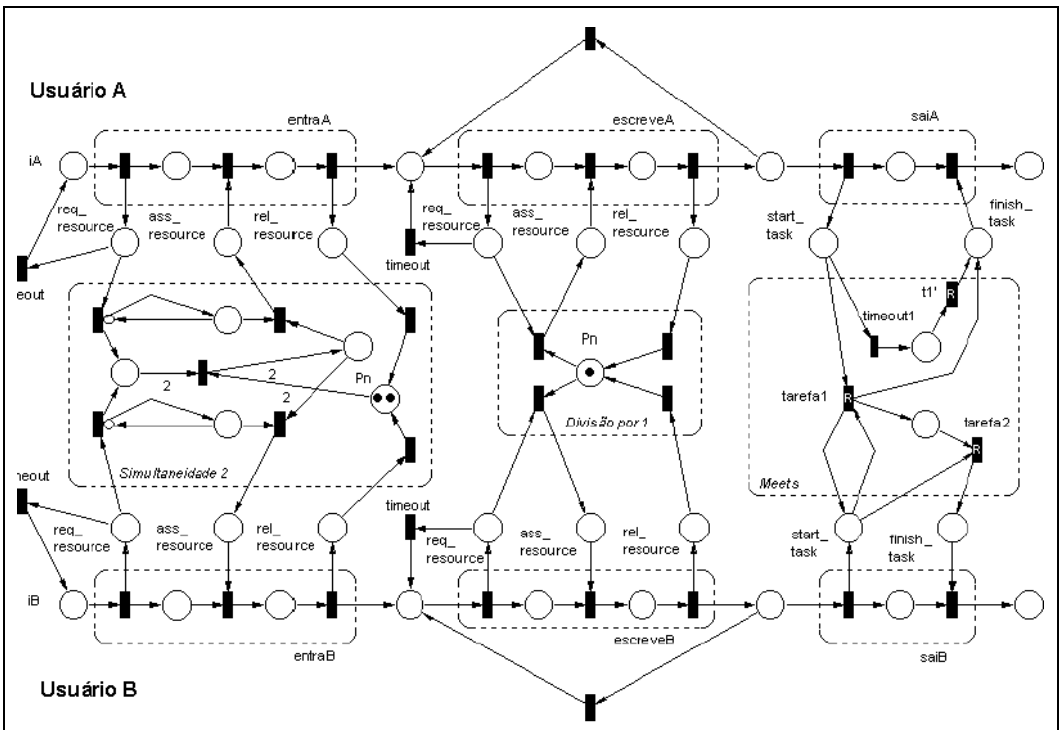


Figura 2-8: Whiteboard – Nível de coordenação.

2.4. A coordenação em Sistemas de Workflow

Sistema de Workflow é uma classe de sistemas CSCW que provê meios para especificação, geração, execução e coordenação de processos organizacionais. Estes processos são determinados por um conjunto de atividades, logicamente

relacionadas, com o objetivo de atingir um determinado resultado [Georgakopoulos 95], [Bull 92].

Promover a cooperação entre os vários atores (humano, computacional, organizacional, entre outros) responsáveis pela execução de atividades que definem os processos, exige um forte componente de coordenação devido a relação de dependência que as atividades, em geral, guardam entre si [Agostine 97]. Equacionar os possíveis conflitos entre as atividades torna-se mais complexo quando o workflow envolve múltiplas organizações [Raposo 00b].

O Sistema de Gerenciamento de Workflow (em inglês, WfMS- Workflow Management System) é exemplo de um sistema que se ocupa em coordenar o workflow resolvendo questões pertinentes às atividades, tais como: qual e quando uma atividade será executada, qual ator deverá assumi-la, quais os recursos necessários [van der Aalst 94]. Uma característica do processo de coordenação de um Workflow é a hipótese da ausência de erros na execução das atividades. Isto é, a execução de uma atividade assume que todo o trabalho realizado anteriormente por outras atividades foi efetuado corretamente [Weber 97]. Um dos problemas encontrado na coordenação é o tratamento de possíveis exceções, i.e. a violação desta hipótese.

A união de empresas e instituições de pesquisas que trabalham na padronização dos conceitos da área de Workflow tem em inglês o acrônimo WfMC (Workflow Management Coalision). No entanto não existe um consenso sobre as técnicas de modelagem tanto dos processos quanto do sistema de gerenciamento. Em [van der Aalst 94] propõe-se um modelo baseado em atividades e usa-se redes de Petri de alto nível (coloridas, temporizadas e hierárquicas) para modelar tanto o workflow quanto o sistema de coordenação do Workflow. Em [Raposo 00b] apresenta-se um conjunto de mecanismos de coordenação, também baseados em redes de Petri, destinados à modelagem das dependências temporais e de recursos entre as atividades que definem o workflow. As razões do uso de redes de Petri na modelagem de sistemas de workflow é justificada em [van der Aalst 96]:

1. a lógica do processo pode ser representada tanto formalmente como graficamente;
2. o estado[7] do processo pode ser modelado explicitamente;
3. uma grande quantidade de técnicas para análise e simulação do Workflow permite verificar a priori a sua correção .

O FlowMark é um exemplo de sistema comercial para gerenciamento de Workflow. Este sistema oferece uma linguagem (FDL) para especificação dos processos que são modelados através de um grafo direcionado acíclico. Os nós desse

[7] Genericamente falando, "o estado de um sistema em um instante de tempo *t* descreve seu comportamento neste momento medido de alguma forma" [Cassandras 93].

grafo correspondem às atividades e os arcos representam o fluxo de controle e os dados entre as atividades conectadas por eles [Alonso 97]. Exemplos de outros produtos comerciais podem ser encontrados em [Marazakis 97].

O comercio eletrônico, a Internet e a tendência de globalização da economia têm impulsionado a cooperação entre as organizações e, conseqüentemente o desenvolvimento de ferramentas para Workflows multi-organizacionais. Em [Verbeek 04] expõe-se algumas das dificuldades encontradas na coordenação e desenvolvimento de Workflows multi-organizacionais e apresenta-se uma ferramenta, baseada em técnicas de análises para redes de Petri, para verificar a correção de workflows multi-organizacionais. Os processos são especificados em uma linguagem baseada em XML que na seqüência é transformada para uma rede de Petri especifica para verificar se o worflow está correto.

O uso de grafos para especificação do Workflow pode ser mais natural para o projetista do que fazê-la diretamente em um sistema como as redes de Petri, por exemplo. Neste caso deve-se ter ferramentas para obter, a partir da especificação, o modelo do workflow no sistema formal desejado. Dessa forma pode-se oferecer facilidades ao projetista e também explorar os benefícios de um sistema formal como as redes de Petri. Uma outra vantagem desta abordagem é poupar o projetista de trabalhos cansativos e repetitivos como o de modelar muitas vezes um mesmo tipo de dependência. Os detalhes de modelagem das dependências podem ser pré-definidos usando o formalismo desejado e inseridos corretamente durante o processo de geração do modelo de Workflow [Cruz 02].

2.5. A coordenação em ambientes Multi-Agentes

Os relacionamentos das pessoas com ambientes de trabalho, familiar, social, etc. geram necessidades e obrigações que devem ser previstas e atendidas eficientemente para evitar-se possíveis conflitos. Uma forma de coordenar estas atividades é contratando os serviços oferecidos por agências especializadas. Por exemplo, agência de despachante, agência de seguro, agência de investimento, agência de viagem, agência imobiliária, etc.

De forma análoga, ambientes computacionais também fazem uso dos serviços oferecidos por agentes[8], sendo que "um agente é qualquer coisa que pode ser vista percebendo seu ambiente através de sensores e atuando neste ambiente através de efetuadores" [Russell 03]. Uma possível representação gráfica para esta definição é ilustrada na Figura 2-9.

[8] Existem muitas definições para agente e que apresentam diferenças entre si conforme a classe do agente (agente móvel, agente inteligente, agente de aprendizagem, etc). [Franklin 96] faz uma comparação entre algumas destas definições.

Para efetuar seus serviços um agente deve ser capaz de "selecionar ações apropriadas em tempo adequado e em uma seqüência correta" [Chen 02]. Estas ações são representadas por tarefas que definem como atingir algum objetivo pretendido. Em um ambiente multi-agente a execução destas tarefas pode afetar ou ser afetada por outras tarefas, o que caracteriza um relacionamento de dependência entre elas.

Se as tarefas pertencem a agentes diferentes então os relacionamentos entre elas apresentam um caráter aqui denominado de dependência não-local. Esta configuração limita a capacidade do agente em selecionar ações apropriadas por não ter conhecimento das restrições derivadas das dependências não-locais [Chen 02].

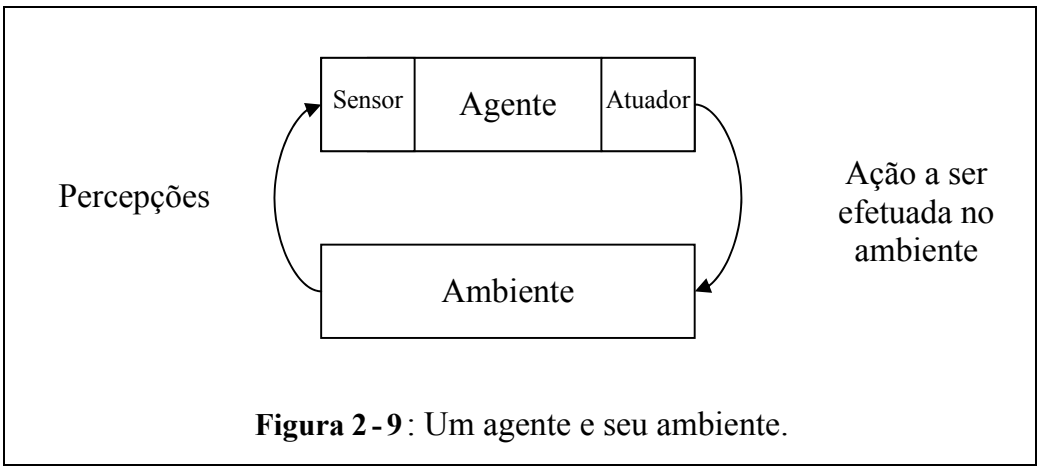


Figura 2 - 9: Um agente e seu ambiente.

O trabalho apresentado em [Holvoet 96] destina-se à especificação de sistemas multi-agentes cooperativos. O objetivo é levar os diferentes agentes a cooperarem entre si na realização de objetivos comuns. Aqui também se observa, embora os agentes sejam autônomos em seus comportamentos, relações de dependências no cumprimento dos seus objetivos. O que se faz é modelar o comportamento de cada agente por uma rede de Petri, separada das ações cooperativas com outros agentes. Em um segundo momento especifica-se as relações de dependência (sincronização) que definem a cooperação entre os agentes. As sincronizações são efetuadas interconectando-se transições pertencentes às redes de Petri dos agentes envolvidos. A rede final resultante deste processo é uma abstração das redes de Petri denominadas redes Genéricas. A metodologia é dependente do problema, i.e. a escolha de quais transições devem ser interconectadas depende de cada caso.

Metodologias baseadas em redes de Petri de alto nível, mais particularmente redes de Petri coloridas, empregadas no problema da coordenação dos comportamentos dos agentes são apresentadas em [Moldt 97] e [Weyns 02].

A tendência em considerar a Internet como uma plataforma de computação distribuída globalmente tem motivado o desenvolvimento de novos modelos de

coordenação. Trabalhos discutindo o potencial das tecnologias de agentes e de coordenação no desenvolvimento de novos modelos computacionais que atendam as necessidades dos serviços oferecidos pela Internet podem ser vistos em [Ciancarini 99] e [Papadopoulos 00].

A situação apresentada nesta seção ilustra uma necessidade, a de modelos de coordenação capazes de fornecer tanto informações locais como globais a respeito do sistema aos agentes e, dessa forma possibilitar-lhes uma seleção apropriada das ações a serem efetuadas no ambiente.

2.6. Considerações do Capítulo

A necessidade de coordenação surge quando existem dependências entre atividades. Estas dependências geram restrições para a execução das atividades, restrições estas que devem ser tratadas de forma a equacionar os conflitos.

Abordar a coordenação de um ambiente computacional considerando a separação entre atividades e dependências, e considerando diferentes níveis de abstração permite encapsular detalhes relacionados às atividades e às dependências e tratá-los em níveis de abstração mais baixos. Esta característica da modelagem traz como benefícios a facilidade em explorar protótipos, analisar e simular a aplicação. Esta abordagem valoriza ainda a implementação de projetos orientados a componentes. Vendo as dependências como componentes, construir novas aplicações consiste em identificar os tipos de dependências entre as atividades e selecionar os componentes adequados [Dellarocas 96], [Raposo 01a].

Este Capítulo situou o processo de coordenação, segundo a definição adotada na Seção 2.1, em alguns campos de pesquisa através de uma breve apresentação e referenciando os principais trabalhos de interesse para o presente contexto. Guiando-se por estes trabalhos, observou-se que o formalismo das redes de Petri oferece uma base sólida e atraente para a modelagem de mecanismos de coordenação, isto em função da sua capacidade de modelar situações que exijam sincronismo e/ou concorrência. Ainda, a abundância de técnicas para análise e simulação permite verificar a correção destes mecanismos antes de implementá-los.

A classe de dependência (causal, recurso, simultaneidade, pré-requisito, produtor-consumidor, etc.) entre atividades varia de acordo com o contexto do problema a ser modelado, no entanto um denominador comum entre estas classes parece ser o fator tempo. Assim, modelos temporais são fundamentais para expressar dependências entre atividades de modo a atender aplicações em diferentes campos de pesquisa [Yoon 98], [Zaidi 99], [Wahl 94].

O Capítulo 3 introduz, a seguir, uma nova metodologia para a modelagem de mecanismos de coordenação. Avança-se no conceito da coordenação, pois além da especificação (identificar/associar) apresenta-se um algoritmo, de custo linear, para

a geração automática de mecanismos de coordenação a partir da especificação dos comportamentos temporais. Estes mecanismos são livres de inconsistências temporais e são globais, isto é, uma atividade está habilitada se não violar qualquer restrição temporal do sistema. Exploram-se as vantagens das redes de Petri na modelagem destes mecanismos sem exigir do projetista do sistema conhecimento deste formalismo. Por fim, permite-se um mapeamento um-para-vários, isto é uma atividade pode relacionar-se com *n* outras atividades simultaneamente.

3. A METODOLOGIA GRAFO DE RELAÇÕES

Os processos de coordenação entre atividades encontrados na literatura modelam o gerenciamento de dependências considerando apenas as atividades inter-relacionadas diretamente, como por exemplo [Dellarocas 96], [Raposo 00a]. No entanto determinados tipos de dependência, por exemplo, as temporais, podem definir restrições a atividades que não estão relacionadas diretamente. O tratamento desses casos, através de ferramentas que modelam apenas as atividades diretamente relacionadas, requer do projetista uma análise laboriosa na identificação de todos os casos e uma posterior verificação se o modelo é consistente também sob a óptica global, ou seja, sob a ótica de todos os relacionamentos.

Oferecer um tratamento para esta questão é a motivação para uma nova metodologia que possibilita expressar analítica e graficamente relações de interdependências entre atividades em um ambiente computacional. Assim gera-se um mecanismo de coordenação que também modela o comportamento global, isto é, uma estrutura que garante a realização de todas as interdependências estabelecidas para o conjunto de atividades.

A dificuldade neste tipo de modelagem está em coordenar eficientemente as dependências entre as atividades de forma a decidir quais e quando elas serão executadas.

Assim, a metodologia proposta, denominada Metodologia **GR**, aborda o problema da coordenação (Seção 3.3) em dois momentos: um global e outro local. A etapa global é responsável por coordenar as condições necessárias para autorização do início de uma atividade. A etapa local coordena a execução entre duas atividades previamente autorizadas obedecendo ao tipo de relação especificado para elas.

Esta abordagem não restringe a quantidade de relacionamentos em que uma atividade pode estar envolvida, isto é, permite que uma atividade se relacione com qualquer número de outras atividades, direta ou indiretamente.

A natureza (temporal, causal, entre outras) da coordenação é estabelecida pelo conjunto de relações permitido entre as atividades. Se estas relações são temporais então se coordena o comportamento das atividades ao longo do tempo (**atividade 1** deve ser executada antes da **atividade 2** e ao mesmo tempo que a **atividade 3** e **atividade 4**, etc.). Se estas relações são causais então se coordena a

execução das atividades baseando-se em eventos (se acontecer **atividade 2** então deve acontecer **atividade 1**, **atividade 3** e **atividade 4**, etc.).

3.1. Conceitos Básicos da Metodologia GR

São apresentados nesta seção alguns dos conceitos usados no desenvolvimento da Metodologia **GR**.

3.1.1. Atividade

As atividades representam as partes funcionais de uma aplicação e têm as seguintes características:

- são atômicas;
- promovem alguma ação no ambiente computacional;
- têm um início e um fim;
- satisfazem as condições impostas para sua execução.

Uma atividade é uma unidade funcional indivisível. O conceito de atividade se aproxima do conceito de transação atômica [Tanenbaum 95]. No entanto, não garante que as alterações efetuadas, no ambiente computacional, em função da execução de uma atividade possam ser desfeitas.

As condições impostas à execução de uma atividade são derivadas das dependências especificadas entre elas e devem ser o resultado de uma análise global. Por exemplo, dadas três atividades **a**, **b** e **c**, se **a** deve ser executada no mesmo intervalo de tempo de **b** e **b** deve ter sua execução iniciada em sincronismo com **c**, por transitividade a condição de sincronização entre **b** e **c** também deve ser aplicada a **a**, isto é, o início de execução de **a**, **b** e **c**, devem ser simultâneos.

Por exemplo, em um Ambiente Virtual Colaborativo, um avatar pode ter várias capacidades, como pular, andar e falar. Cada uma destas capacidades pode ser modelada como uma atividade. Considerando uma animação 3D, onde os atores são dançarinos e executam diferentes passos, cada um destes passos pode ser uma atividade. Em uma apresentação multimídia, a reprodução de um segmento de vídeo, a reprodução de um áudio ou a apresentação de um texto podem ser atividades. A Definição 1 formaliza o conceito de atividade.

Definição 1: atividade

> Uma **atividade** é uma ação atômica efetuada em um ambiente computacional durante um intervalo de tempo finito e não nulo, cuja execução é autorizada mediante o cumprimento de um conjunto de restrições previamente definidas.

Definição 2: evento

Um evento é um conceito primitivo, com duração nula, e associado ao início ou fim de uma atividade.

O evento início da execução de uma atividade **a** pode ocorrer:

1. através da interação com o usuário do sistema;
2. através da especificação do projetista do sistema;
3. por solicitação de uma atividade **b**;
4. em função do início ou fim de uma atividade **b**.

O evento fim da execução de uma atividade **a** é determinado:

1. após despender o intervalo de tempo associado à atividade **a**.

A definição de atividade formulada acima não determina como ela deve ser executada. Isto torna seu conceito independente do domínio do problema permitindo seu emprego na especificação de aplicações em diferentes domínios.

3.1.2. Modelagem Temporal

As atividades de uma aplicação acontecem (são executadas) no tempo, isto é, cada uma consome um intervalo finito de tempo para ser efetuada. O tempo torna-se um parâmetro razoável para estabelecer relacionamentos entre elas. Isto permite uma primeira abstração: a associação de uma atividade a um intervalo finito de tempo.

Uma atividade pode acontecer em paralelo com um conjunto de outras atividades e/ou em seqüência com outras. A necessidade de integrá-las segundo suas dependências temporais pode ser equacionada por um Modelo Temporal. Este deve possibilitar a representação de informações temporais sob dois aspectos: estabelecer quando as atividades devem acontecer e garantir que as restrições temporais estabelecidas entre elas sejam viáveis.

Dentre as características desejadas para este formalismo destacam-se a sua capacidade de expressão (representação) e de especificação dos relacionamentos entre as atividades (sistema).

Segundo [Wahl 94], em geral os formalismos com grande capacidade de expressão estão apoiados em lógicas temporais, cuja manipulação é complexa, não oferecendo uma semântica intuitiva a usuários inexperientes em especificações formais.

Outros questionamentos dizem respeito ao tipo de inferências que podem ser obtidas analisando-se as expressões resultantes da especificação dos relacionamentos temporais entre as atividades como por exemplo:

- estas expressões são livres de inconsistências?
- se for necessário detectá-las após a especificação, qual o custo?

Em [Zaidi 99], é apresentada uma metodologia para formalizar aspectos temporais de sistemas a eventos discretos. O modelo atende a um conjunto de axiomas propostos que permitem inferir novos relacionamentos entre as atividades do sistema. As expressões obtidas na representação do sistema podem ser inconsistentes e a detecção delas exige um cômputo extra.

Classificação dos Modelos Temporais

Existe uma grande variedade de ferramentas matemáticas empregadas na construção dos modelos temporais nas diferentes áreas das Ciências e das Engenharias. Estes modelos, em geral, trabalham com dois tipos de abordagem ([Wahl 94]), modelos baseados em eventos e modelos baseados em atividades.

Os **modelos baseados em eventos - ME** trabalham, como sugere o próprio nome com eventos, pontos, isto é, instantes de tempo, os quais podem representar um evento qualquer do sistema. Em geral eventos estão associados a informações de causalidades, por exemplo o início de uma atividade ([Duda 95], [Duda 97]).

As informações temporais de um sistema, modeladas por um **ME**, são representadas através de relações que trabalham com as posições relativas de dois pontos (eventos) arbitrários no espaço do tempo. Isto é, um evento **p** pode acontecer "antes" (before) de um evento **q**, um evento **p** pode acontecer "simultaneamente" (equals) a um evento **q** e por fim, um evento **p** pode acontecer "depois" (after) de um evento **q** (Figura 3 - 1).

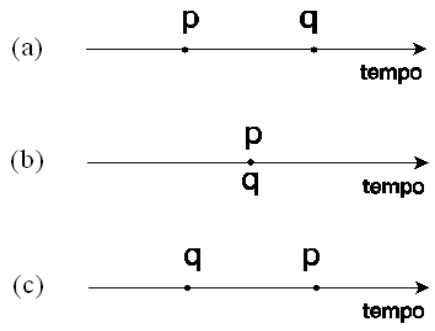


Figura 3 - 1 : Relações entre eventos **(a)** p **before** q, **(b)** p **equals** q, **(c)** p **after** q.

Os **modelos baseados em atividades - MA** têm como elemento fundamental intervalos não nulos. Estes intervalos contêm as atividades que são executadas no sistema, as quais por sua vez podem estabelecer relacionamentos entre si por meio de um ordenamento temporal dos intervalos associados a elas.

Definição 3: ordenamento temporal

> Um ordenamento temporal é um conjunto de regras que permite por em ordem intervalos de tempo em um espaço temporal.

Definição 4: intervalo de tempo na metodologia GR

> Um intervalo de tempo é um intervalo fechado à esquerda e aberto à direita, $\mathbf{a} = [\mathbf{a}_i, \mathbf{a}_f)$, $\mathbf{a}_i < \mathbf{a}_f$ e $\mathbf{a}_i, \mathbf{a}_f \in \Re^+$.

A álgebra de intervalos apresentada em [Allen 84], estabelece um ordenamento temporal considerando todas as combinações entre dois intervalos de tempo não nulos **x** e **y**. Estes intervalos podem ser posicionados sobre a linha do tempo de sete formas distintas (ver Figura 3 - 2), determinando um conjunto de relações temporais possíveis entre duas atividades **x** e **y**.

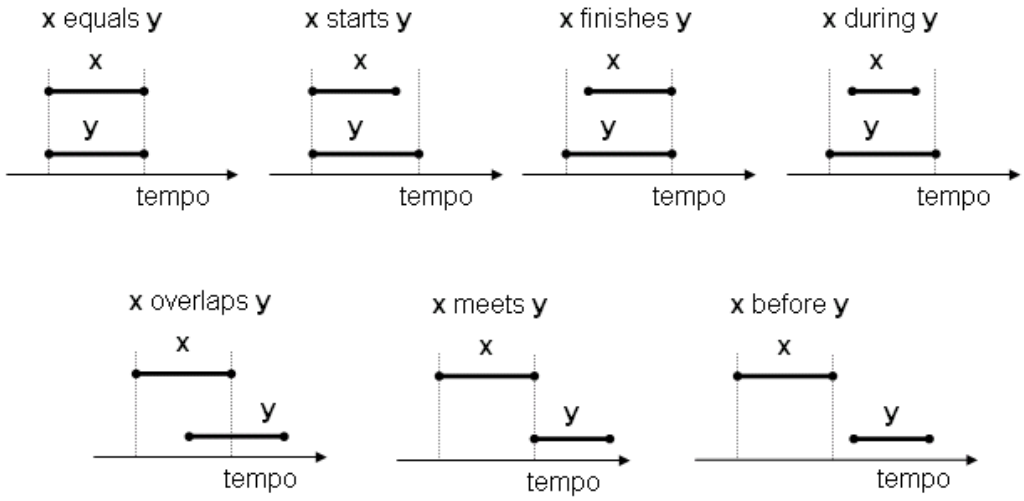


Figura 3 - 2 : Relações temporais entre intervalos apresentadas em **[Allen 84]**

A seguir apresentam-se as relações temporais entre intervalos de tempo aqui denominadas **P7** [Allen 84]:

x before y → x deve ser efetuada antes de y.

x equals y → x deve ser efetuada no mesmo intervalo de tempo de y

x meets y → y deve ser efetuada imediatamente após o final x, ou seja, não existe intervalo de tempo entre x e y.

x overlaps y → x deve começar antes de y que deve começar antes de x terminar e x termina antes de y.

x during y → x deve começar depois de y e deve terminar antes de y.

x starts y → x e y devem começar juntas e **x** deve terminar antes de y.

x finishes y → x e y devem terminar juntas e **x** deve começar depois de y**.**

As relações temporais **P7** satisfazem ainda as propriedades de mútua exclusão **p1** e mútua exaustão **p2**. Estas propriedades são enunciadas a seguir para as atividades x e y:

p1) Se $\mathbf{x}\ \mathbf{r_i}\ \mathbf{y}$, onde $\mathbf{r_i} \in \mathbf{P7}$ então não existe $\mathbf{r_j} \neq \mathbf{r_i}, \mathbf{r_j} \in \mathbf{P7}$, tal que $\mathbf{x}\ \mathbf{r_j}\ \mathbf{y}$ seja verdadeiro (exclusão mútua);

p2) Dados $\mathbf{x}$ e $\mathbf{y}$ existe $\mathbf{r_i} \in \mathbf{P7}$ tal que $\mathbf{x}\ \mathbf{r_i}\ \mathbf{y}$ ou $\mathbf{y}\ \mathbf{r_i}\ \mathbf{x}$ é verdadeira[9].

Alguns modelos propostos [Duda 95], [Zaidi 99] são baseados nos dois elementos: ponto e intervalo. Estes modelos podem ser classificados como **ME**, pois apresentam as mesmas características dos modelos baseados em eventos, isto é, podem ser traduzidos para um modelo **ME** [Wahl 94].

3.1.3. Mecanismos de Coordenação

A coordenação pode ser vista como o processo de administrar dependências entre atividades [Malone 94].

No presente trabalho, a coordenação surge da necessidade de administrar dependências entre atividades de modo a decidir qual atividade será executada e quando será executada. Esta decisão deve garantir a integridade das dependências entre as atividades: uma atividade **x** relacionada com uma atividade **y** só é executada se as dependências entre elas forem completamente satisfeitas.

Para estabelecer esta coordenação em um ambiente computacional utiliza-se um Mecanismo de Coordenação. O conceito de Mecanismo de Coordenação em um ambiente computacional segundo [Schimidt 96]: "deve ser idealizado como um

[9] A primitiva equals é uma exceção pois x equals y ≡ y equals x .

dispositivo de software especializado, o qual interage com uma aplicação de software específica e apóia o trabalho de articulação (coordenação) com relação ao campo de trabalho da maneira como representado pelas estruturas de dados e funcionalidades da referida aplicação". Seguindo este conceito, a Definição 5 formaliza este mecanismo para o contexto da metodologia apresentada neste texto.

Definição 5: mecanismo de coordenação

> Um Mecanismo de Coordenação **MC** é um dispositivo de software que interage com uma aplicação de software modelando e coordenando as dependências entre suas atividades de forma a garantir que sejam respeitadas.

A metodologia discutida neste trabalho propõe três níveis de abstração para a modelagem de sistemas conforme a Figura 3 - 3.

Figura 3 - 3: Níveis de abstração **N1**, **N2** e **N3**.

No nível da especificação **N1**, ou **espaço das expressões**, são modelados os comportamentos do sistema, representados por meio das relações estabelecidas entre as atividades. Em **N1** diz-se **quais** são as atividades e as dependências temporais entre elas.

Os Mecanismos de Coordenação **MC** são construídos no nível da coordenação **N2**, ou **espaço dos mecanismos de coordenação**. Estas estruturas são responsáveis por estabelecer um mecanismo cujo comportamento dinâmico atende |s especificações expressas em **N1**. Este mecanismo é o fundamento para a implementação do coordenador no nível **N3**. O **MC** é o resultado da identificação e modelagem de todas as restrições temporais derivadas das dependências entre as atividades descritas em **N1**.

Finalmente, no nível da execução **N3**, um programa denominado **coordenador** deve implementar o **MC** obtido em **N2**. Este coordenador interage com o conjunto de atividades obedecendo a especificação efetuada em **N1**.

Os detalhes de implementação do ambiente de execução das atividades e do programa que implementa o **MC** não serão tratados neste trabalho.

Nas próximas seções apresentam-se o formalismo proposto para a metodologia **GR**, usado no nível **N1**, e os procedimentos e modelos matemáticos empregados na construção dos Mecanismos de Coordenação, usados no nível **N2**.

3.2. Modelagem Temporal em GR

A metodologia Grafo de Relações, ou simplesmente metodologia **GR**, propõe-se a modelar e coordenar comportamentos de um sistema **S** que possa ser descrito por dependências temporais entre suas atividades. Mais especificamente o problema equacionado pela metodologia **GR** é descrito a seguir.

Problema:

> Construir a partir da especificação do comportamento temporal de um sistema **S**, definido por seu conjunto de atividades $\mathbf{A} = \{\mathbf{a}_1, \mathbf{a}_2, \cdots, \mathbf{a}_n\}$, um **MC** que coordene os comportamentos de **S** de modo a determinar quando e qual atividade pode ser executada.

A abordagem para a solução do problema proposto implicará na definição de dois modelos: o modelo temporal no nível de abstração da especificação **N1** e o modelo de coordenação no nível de abstração **N2**.

O modelo temporal proposto em **N1** é baseado em intervalos de tempo (Definição 4) e emprega o conceito de grafos na representação das expressões que descrevem os comportamentos de **S**. Isto proporciona à metodologia **GR** tanto uma representação analítica como uma representação gráfica para os comportamentos.

Em **N2**, implementam-se os comportamentos temporais especificados em **N1** obtendo-se como resultado o mecanismo de coordenação - **MC**. O processo de construção do **MC** permite visualizá-lo como sendo a composição de dois mecanismos de coordenação que trabalham de forma integrada: o Mecanismo de Coordenação Global - **MCG**, que se concentra em identificar e modelar as restrições temporais (Definição 7) impostas para a execução das atividades, e os Mecanismos de Coordenação Local – **MCL**, que implementam os dispositivos de coordenação para cada par de atividades que mantém algum tipo de dependência temporal.

As Seções 3.3.1 e 3.3.2 desenvolvem o modelo temporal.

3.2.1. O Conjunto de Relações Temporais na metodologia GR

O conjunto de relações definido para metodologia **GR** tem por objetivo formalizar as dependências temporais que podem ser especificadas entre as atividades da aplicação.

Considerando o fato de duas atividades acontecerem ou em paralelo ou em seqüência e a ordem temporal apresentada por [Allen 84], adota-se desta álgebra de intervalos o conjunto de sete primitivas (ilustrados na Figura 3 - 2) como sendo as possibilidades de relacionamento entre duas atividades.

Definição 6: conjunto de relações temporais na metodologia GR

Sejam **a** e **b** atividades distintas que ocorrem nos intervalos de tempo $\mathbf{x} = [\mathbf{a_i}, \mathbf{a_f})$ e $\mathbf{y} = [\mathbf{b_i}, \mathbf{b_f})$, respectivamente, sendo $\mathbf{a_i}$ e $\mathbf{a_f}$, $\mathbf{b_i}$ e $\mathbf{b_f}$ o início e o fim de **a** e **b** respectivamente. O conjunto de relações temporais binárias **D** é dado por:

$$\mathbf{D} = \{\mathbf{e},\ \mathbf{s},\ \mathbf{d},\ \mathbf{f},\ \mathbf{o},\ \mathbf{m},\ \mathbf{b}\}, \text{ onde}$$

$\mathbf{e}(\mathbf{a},\mathbf{b}) \leftrightarrow \mathbf{a_i} = \mathbf{b_i}$ e $\mathbf{a_f} = \mathbf{b_f}$
a é efetuada no mesmo intervalo de tempo de **b**

$\mathbf{s}(\mathbf{a},\mathbf{b}) \leftrightarrow \mathbf{a_i} = \mathbf{b_i}$ e $\mathbf{a_f} < \mathbf{b_f}$
a e **b** começam juntas, mas **a** termina antes de **b**.

$\mathbf{d}(\mathbf{a},\mathbf{b}) \leftrightarrow \mathbf{a_i} > \mathbf{b_i}$ e $\mathbf{a_f} < \mathbf{b_f}$
a começa depois de **b** e termina antes de **b**.

$\mathbf{f}(\mathbf{a},\mathbf{b}) \leftrightarrow \mathbf{a_i} > \mathbf{b_i}$ e $\mathbf{a_f} = \mathbf{b_f}$
a começa depois de **b**, mas **a** e **b** terminam juntas.

$\mathbf{o}(\mathbf{a},\mathbf{b}) \leftrightarrow \mathbf{a_i} < \mathbf{b_i}$ e $\mathbf{b_i} < \mathbf{a_f}$ e $\mathbf{a_f} < \mathbf{b_f}$
a começa antes de **b** que começa antes de **a** terminar. **a** termina antes de **b**.

$\mathbf{m}(\mathbf{a},\mathbf{b}) \leftrightarrow \mathbf{a_f} = \mathbf{b_i}$
b é efetuada imediatamente após o final de **a**.

$\mathbf{b}(\mathbf{a},\mathbf{b}) \leftrightarrow \mathbf{a_f} < \mathbf{b_i}$
a deve ser efetuada antes de **b**.

Os símbolos **e**, **s**, **d**, **f, o**, **m** e **b** indicam as primitivas **equals**, **starts**, **during**, **finishes**, **overlaps**, **meets** e **before**, respectivamente. Estas duas notações serão usadas indistintamente no texto, dando-se preferência para aquela que for mais

conveniente para a situação em questão. Uma relação **r** de **D** é definida por um par ordenado de atividades $(\mathbf{a},\mathbf{b})$, isto é $\mathbf{r}(\mathbf{a},\mathbf{b})$. A relação inversa será denotada por $\mathbf{r}^{-1}$, isto é $\mathbf{r}^{-1}(\mathbf{a},\mathbf{b})$. No decorrer do texto, ao se fazer referências a uma relação **r** também estará sendo considerada implicitamente sua inversa e se necessário serão feitas as diferenciações devidas.

Definição 7: restrição temporal

> Uma Restrição Temporal é uma equação ou inequação derivada das relações temporais do conjunto **D.**

3.2.2. As expressões na metodologia GR

Uma atividade $\mathbf{a_i}$ pode relacionar-se com um número arbitrário **k** de outras atividades $\mathbf{b_1},\mathbf{b_2},\cdots,\mathbf{b_k}$ por meio das primitivas definidas em **D**. Para tal é necessário estabelecer **k** relacionamentos, devido a natureza binária das primitivas. Por exemplo, os **k** relacionamentos poderiam ser $(\mathbf{a_i}\ \textbf{starts}\ \mathbf{b_1})$, $(\mathbf{a_i}\ \textbf{starts}\ \mathbf{b_2})$, $(\mathbf{b_3}\ \textbf{before}\ \mathbf{a_i})$, $\cdots$, $(\mathbf{a_i}\ \textbf{during}\ \mathbf{b_k})$.

A Figura 3 - 4 ilustra uma faixa do espectro de possibilidades para $\mathbf{k=2}$, considerando as relações $\mathbf{r_1}$, $\mathbf{r_2}$ e as atividades **x**, **y** e **z**. A atividade **y** relaciona-se com **x** e **z**. A relação $\mathbf{r_1}$ entre **x** e **y** é representada pelo eixo vertical e a relação $\mathbf{r_2}$ entre **y** e **z** é representada pelo eixo horizontal.

A representação dos comportamentos temporais entre **x**, **y** e **z** pode ser expressa por uma seqüência das três atividades intercaladas com as relações: $\mathbf{x}\ \mathbf{r_1}\ \mathbf{y}\ \mathbf{r_2}\ \mathbf{z}$. Por exemplo, para $\mathbf{r_1}=\textbf{starts}$ e $\mathbf{r_2}=\textbf{meets}$ tem-se a expressão $\mathbf{x}\ \textbf{starts}\ \mathbf{y}\ \textbf{meets}\ \mathbf{z}$. No entanto esta forma de expressão torna-se inviável para **k**>2.

Assim para estes casos (**k**>2) uma escolha é descrever os relacionamentos entre as atividades usando um grafo. A Teoria dos Grafos é a base de um modelo matemático apropriado para atender os requisitos expostos e oferecer uma estrutura sólida e elegante para o processo de formalização da metodologia **GR**, o qual será abordado nas próximas seções deste capítulo.

Feita esta escolha, passa-se a representar os comportamentos temporais de **S** através de um grafo $\mathbf{E}(\mathbf{A},\mathbf{R},\mathbf{F})$, conforme estabelecem a Definição 8 e a Definição 9.

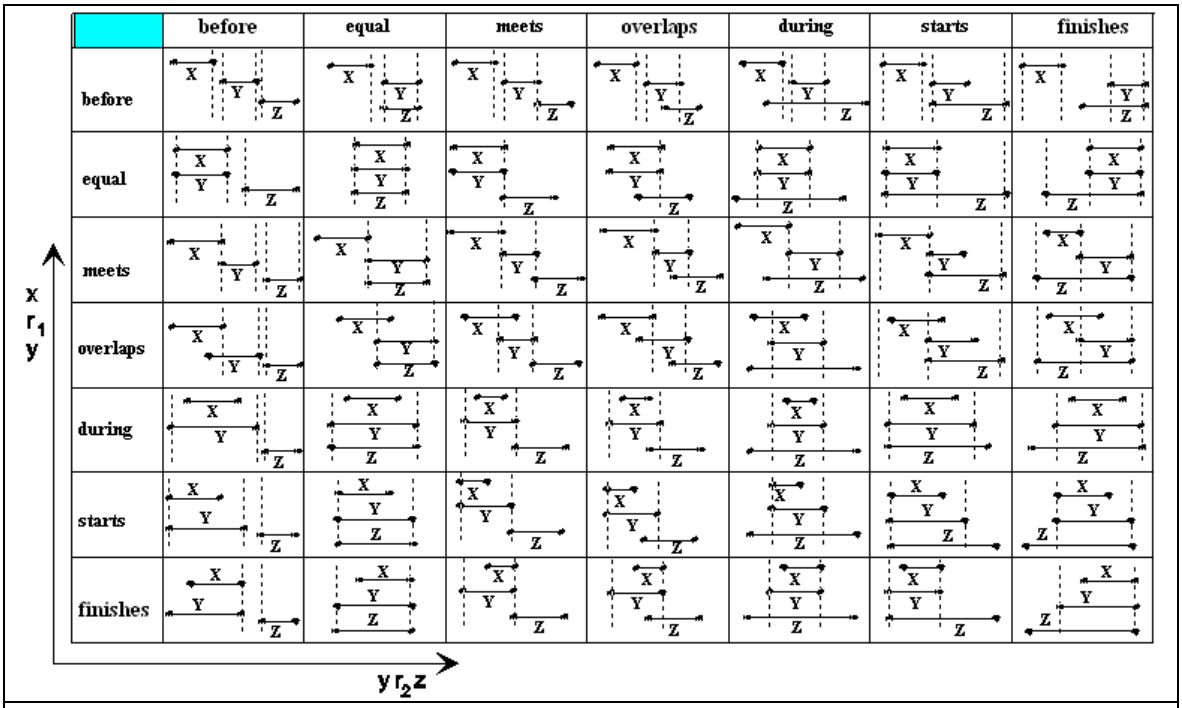


Figura 3 - 4 : Espectro parcial das relações entre as atividades **x** , **y** e **z** .

Definição 8: expressão na metodologia GR

Uma expressão $\mathbf{E(A,R,F)}$ é um grafo direcionado, rotulado[10] e com grafo subjacente[11] acíclico, onde **A** é um conjunto de vértices que representam as atividades, $\mathbf{R \subset A \times A,}$ é um conjunto de arestas e **F** é uma função $\mathbf{F : R \rightarrow \wp(D)}$, onde $\mathbf{\wp(D)}$ é o conjunto das partes de **D**, denominada função rotuladora de arestas.

A função **F** associa cada par ordenado $\mathbf{(a_i, a_j) \in R}$ a um subconjunto de **D**, sendo **D** o conjunto das relações definidas na metodologia **GR** conforme a Definição 6 .

Um exemplo de expressão é dado por:

$$\mathbf{A = \{a_1, a_2, a_3, a_4\},}$$
$$\mathbf{R = \{(a_1, a_2), (a_3, a_1), (a_1, a_4)\}}$$

e **F** dada pela seguinte lei de associação:

[10] "Um grafo é denominado rotulado em vértices (ou arestas) quando a cada vértice (ou aresta) estiver respectivamente associado um conjunto, denominado *rótulo*." [Szwarcfiter 86]. Neste texto considera-se um grafo rotulado em vértices e em arestas, denominado *grafo rotulado*.

[11] O grafo subjacente é aquele obtido do grafo direcionado retirado os sentidos das arestas [Szwarcfiter 86].

$$\mathbf{F}(\mathbf{a}_1, \mathbf{a}_2) = \{\mathbf{d}\};$$
$$\mathbf{F}(\mathbf{a}_3, \mathbf{a}_1) = \{\mathbf{e}, \mathbf{s}, \mathbf{f}\};$$
$$\mathbf{F}(\mathbf{a}_1, \mathbf{a}_4) = \{\mathbf{s}\}.$$

Dessa forma define-se a expressão $\mathbf{E1}(\mathbf{A}, \mathbf{R}, \mathbf{F})$.

Por simplicidade passa-se a denotar o rótulo $\mathbf{F}(\mathbf{a}_i, \mathbf{a}_j)$ por $\mathbf{Y}_{i,j}$.

A associação dos elementos de **R** a $\wp(\mathbf{D})$ é definida para cada caso de interesse num espectro de $(2^7)^{\mathbf{c}}$ possibilidades, onde **c** é o número de arestas do grafo. Por exemplo, considerando **E1** existem $(2^7)^3$ associações possíveis para **F**, em particular foram consideradas de interesse as seguintes:

$$\mathbf{Y}_{1,2} = \mathbf{F}(\mathbf{a}_1, \mathbf{a}_2) = \{\mathbf{d}\};$$
$$\mathbf{Y}_{3,1} = \mathbf{F}(\mathbf{a}_3, \mathbf{a}_1) = \{\mathbf{e}, \mathbf{s}, \mathbf{f}\} \text{ e}$$
$$\mathbf{Y}_{1,4} = \mathbf{F}(\mathbf{a}_1, \mathbf{a}_4) = \{\mathbf{s}\}.$$

No grafo, o sentido da aresta $(\mathbf{a}_i, \mathbf{a}_j)$ indica a ordem em que as atividades $\mathbf{a}_i$ e $\mathbf{a}_j$ estão relacionadas. Por exemplo, $\mathbf{F}(\mathbf{a}_i, \mathbf{a}_j) = \{\mathbf{s}\}$ significa $\mathbf{a}_i$ **starts** $\mathbf{a}_j$ que é diferente de $\mathbf{a}_j$ **starts** $\mathbf{a}_i$.

O rótulo de uma aresta $(\mathbf{a}_i, \mathbf{a}_j)$, determinado por **F**, indica a relação entre duas atividades $\mathbf{a}_i$ e $\mathbf{a}_j$. Se o rótulo é um conjunto com mais de um elemento então $\mathbf{a}_i$ e $\mathbf{a}_j$ podem relacionar-se selecionando (no nível **N3**) uma das primitivas listadas. Por exemplo, $\mathbf{F}(\mathbf{a}_i, \mathbf{a}_j) = \{\mathbf{e}, \mathbf{s}, \mathbf{f}\}$, significa que qualquer uma das três alternativas **e**, **s** ou **f** poderá ser escolhida durante a execução da aplicação. Em outras palavras o Mecanismo de Coordenação produzido no nível de abstração **N2** deverá permitir, ao coordenador implementado no nível **N3**, que qualquer uma das três alternativas possa ocorrer, cada vez que as atividades $\mathbf{a}_i$ e $\mathbf{a}_j$ forem ativadas pela aplicação.

Uma extensão da Definição 8 pode ser produzida permitindo-se definir alternativas entre as arestas do grafo da expressão $\mathbf{E}(\mathbf{A}, \mathbf{R}, \mathbf{F})$. Esta extensão é útil em situações nas quais não haja necessidade de se respeitar simultaneamente todas as arestas ligadas a um mesmo vértice.

Assim, a Definição 9 introduz a função rotuladora **G** que permite associar, a um vértice de uma expressão, um conjunto de vértices adjacentes a este. O valor que a função **G** assume no vértice é chamado de rótulo da atividade. A semântica desta função **G** está associada à subtração no grafo de todas, exceto uma das arestas que se

ligam ao vértice em questão definidos pela função. Observe-se que os vértices não incluídos no rótulo da atividade não têm suas arestas afetadas por **G**.

Definição 9: expressão qualificada na metodologia GR

Uma expressão qualificada é um par $(\mathbf{E},\mathbf{G})$, onde **E** é uma expressão $\mathbf{E}(\mathbf{A},\mathbf{R},\mathbf{F})$ e **G** é uma função $\mathbf{G}:\mathbf{A}\rightarrow\wp(\mathbf{A})$, onde $\wp(\mathbf{A})$ é o conjunto das partes de **A** denominada função rotuladora de vértices, que satisfaz as seguintes propriedades:

i. Se $\mathbf{b}\in\mathbf{G}(\mathbf{a})$ então a aresta definida por **a** e **b** pertence a **R**;
ii. Se $\mathbf{b}\in\mathbf{G}(\mathbf{a})$ então $\mathbf{a}\notin\mathbf{G}(\mathbf{b})$;
iii. Se $\mathbf{b}\in\mathbf{G}(\mathbf{a})$ então $\exists\,\mathbf{c}\in\mathbf{G}(\mathbf{a})\,|\,\mathbf{c}\neq\mathbf{b}$.

Em outras palavras, a função **G** associa a cada elemento $\mathbf{a_i}$ de **A** um elemento do conjunto das partes de **A**, denominado rótulo de $\mathbf{a_i}$, de tal forma que:

i. Se **b** está no rótulo de **a** então existe uma relação entre **a** e **b**;
ii. Se **b** está no rótulo de **a** então **a** não está no rótulo de **b**;
iii. O rótulo de uma atividade tem pelo menos dois elementos.

Um exemplo de expressão qualificada é dado considerando-se a expressão $\mathbf{E1}(\mathbf{A},\mathbf{R},\mathbf{F})$ definida anteriormente e a função **G** apresentada a seguir:

$$\mathbf{G1}(\mathbf{a_1})=\{\mathbf{a_2},\mathbf{a_4}\}\text{ e}$$
$$\mathbf{G1}(\mathbf{a_2})=\mathbf{G1}(\mathbf{a_3})=\mathbf{G1}(\mathbf{a_4})=\varnothing;$$

que dessa forma define a expressão qualificada $(\mathbf{E1},\mathbf{G1})$.

Novamente, por praticidade passa-se a denotar o rótulo $\mathbf{G}(\mathbf{a_i})$ por $\mathbf{X_i}$.

Assim, o rótulo de uma atividade $\mathbf{a_i}$ indica uma seleção (que acontece no nível **N3**) entre as atividades constantes do rótulo. Por exemplo, $\mathbf{X_i}=\{\mathbf{a_j},\mathbf{a_h},\mathbf{a_k}\}$, indica que somente uma das atividades será selecionada para estabelecer uma relação com $\mathbf{a_i}$, conforme a Definição 9. Esta seleção ocorre cada vez que a atividade $\mathbf{a_i}$ e uma outra atividade pertencente ao seu rótulo forem ativadas, através de eventos (Definição 2) pela aplicação (nível **N3**). Esta funcionalidade deve ser viabilizada pelo Mecanismo de Coordenação produzido no nível de abstração **N2**.

No decorrer do texto, será usado o termo expressão, denotado por ε, referenciando tanto a expressão $\mathbf{E}(\mathbf{A},\mathbf{R},\mathbf{F})$ quanto a expressão qualificada $(\mathbf{E},\mathbf{G})$. Quando necessário far-se-á uso do termo específico.

A representação gráfica de uma expressão em **GR** corresponde à representação gráfica de um grafo, onde as atividades $\mathbf{a_i}$, $\mathbf{i}=\mathbf{1,2,}\cdots\mathbf{,n}$ são

simbolizadas por pontos distintos do plano e em posições arbitrárias. Os elementos de **R** correspondem às setas que conectam dois pontos distintos do plano dado pelo par ordenado. As setas pontilhadas indicam uma operação de seleção. Por convenção os rótulos representados pelo conjunto vazio não são descritos no gráfico. Por exemplo, seja a expressão qualificada $(\mathbf{E2},\mathbf{G2})$ e $\mathbf{E2}(\mathbf{A},\mathbf{R},\mathbf{F2})$, sendo:

$$\mathbf{A} = \{\mathbf{a}_1,\mathbf{a}_2,\mathbf{a}_3,\mathbf{a}_4,\mathbf{a}_5,\mathbf{a}_6,\mathbf{a}_7,\mathbf{a}_8,\mathbf{a}_9,\mathbf{a}_{10},\mathbf{a}_{11},\mathbf{a}_{12}\};$$

$$\mathbf{R} = \left\{\begin{matrix}(\mathbf{a}_1,\mathbf{a}_3),\ (\mathbf{a}_3,\mathbf{a}_2),(\mathbf{a}_3,\mathbf{a}_{11}),(\mathbf{a}_{11},\mathbf{a}_{12}),(\mathbf{a}_{11},\mathbf{a}_{10}),(\mathbf{a}_9,\mathbf{a}_{11}),\\ (\mathbf{a}_{11},\mathbf{a}_6),\ (\mathbf{a}_7,\mathbf{a}_6),(\mathbf{a}_7,\mathbf{a}_5),(\mathbf{a}_8,\mathbf{a}_7),(\mathbf{a}_8,\mathbf{a}_4)\end{matrix}\right\};$$

F2 a função rotuladora de arestas dada por:

$$\mathbf{Y}_{1,3} = \{\mathbf{f}\},\ \mathbf{Y}_{3,2} = \{\mathbf{b}\},\ \mathbf{Y}_{3,11} = \{\mathbf{f}\},\ \mathbf{Y}_{11,12} = \{\mathbf{o}\},$$
$$\mathbf{Y}_{11,10} = \{\mathbf{e}\},\ \mathbf{Y}_{9,11} = \{\mathbf{s},\mathbf{d},\mathbf{f}\},\ \mathbf{Y}_{11,6} = \{\mathbf{s}\},\ \mathbf{Y}_{7,6} = \{\mathbf{s}\},$$
$$\mathbf{Y}_{7,5} = \{\mathbf{m}\},\ \mathbf{Y}_{8,7} = \{\mathbf{b}\} \text{ e } \mathbf{Y}_{8,4} = \{\mathbf{m}\} \text{ e}$$

G2 a função rotuladora de atividades dada por:

$$\mathbf{X}_{11} = \{\mathbf{a}_{12},\mathbf{a}_{10},\mathbf{a}_9\} \text{ e}$$
$$\mathbf{X}_1 = \mathbf{X}_2 = \cdots = \mathbf{X}_{10} = \mathbf{X}_{12} = \varnothing.$$

A Figura 3-5 ilustra a representação gráfica de $(\mathbf{E2},\mathbf{G2})$.

O grau de uma atividade $\mathbf{a}_i$, denotado por $\partial(\mathbf{a}_i)$, é o número de atividades $\mathbf{a}_j$, $\mathbf{i} \neq \mathbf{j}$, relacionadas a ela.

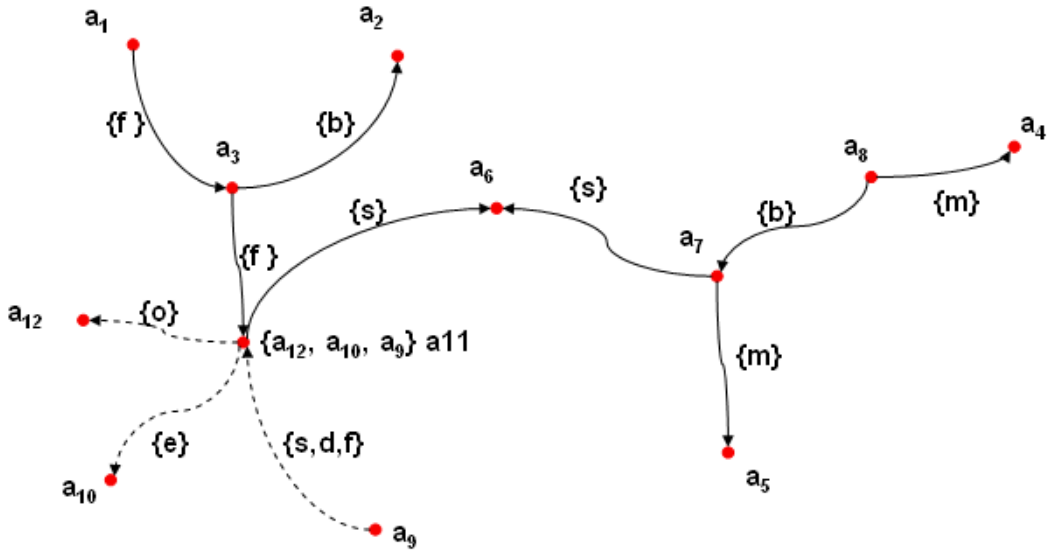


Figura 3-5: Representação gráfica de $(\mathbf{E2},\mathbf{G2})$.

Conforme a Definição 8 tem-se que, uma expressão é cíclica quando existe uma seqüência de $\mathbf{k}$, $\mathbf{1 \le k < n}$, atividades inter-relacionadas, partindo de uma atividade $\mathbf{a_i \in A}$ e voltando em $\mathbf{a_i}$, caso contrário a expressão é acíclica. Neste documento investigam-se as expressões acíclicas. Um caso particular de expressão acíclica é dado pela Definição 10.

Definição 10: expressão linear

Uma expressão linear é uma expressão *acíclica* cujo grau de qualquer uma de suas atividades $\mathbf{a_i}$ satisfaz a seguinte propriedade: $\mathbf{0 < \partial(a_i) \le 2}$.

A fim de motivar a apresentação do resultado a ser estabelecido no Lema 1 e Teorema 1 considere-se a expressão cíclica $\mathbf{E3(A, R, F3)}$, onde:

$\mathbf{A = \{a_1, a_2, a_3\}}$,

$\mathbf{R = \{(a_1, a_2), (a_2, a_3), (a_3, a_1)\}}$ e

F3 a função rotuladora de arestas definida por:

$$\mathbf{Y_{1,2} = \{s\}, Y_{2,3} = \{s\}, Y_{3,1} = \{s\}}$$

e o conceito de consistência apresentado na Definição 11.

Definição 11: expressão consistente[12]

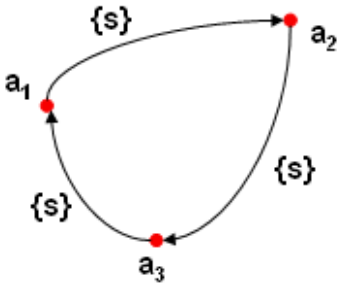
Uma expressão é consistente se todas as restrições temporais derivadas das dependências temporais forem satisfeitas.

As restrições temporais derivadas da Definição 6 são equações e/ou inequações formadas por um operador $<, =$ ou $>$ em seu sentido convencional e operandos **x** e **y** com valores no conjunto dos reais $\Re$ representando os tempos de início e/ou fim das atividades. Portanto o conjunto de restrições temporais de uma expressão é inconsistente se existir pelo menos um par de restrições da forma (1), (2) ou (3):

(1) $\mathbf{x < y}$ e $\mathbf{x > y}$;
(2) $\mathbf{x = y}$ e $\mathbf{x < y}$
(3) $\mathbf{x = y}$ e $\mathbf{x > y}$

[12] Esta definição próxima de sua similar na lógica clássica, como por exemplo em [Mates 72]: "Um conjunto de sentenças Γ é consistente se e somente se existe uma interpretação sob a qual todas as sentenças de Γ são verdadeiras".

A expressão **E3** está representada graficamente na Figura 3 - 6(a) juntamente com as restrições temporais derivadas do seu conjunto de primitivas (Figura 3 - 6(b)). Verificando-se este conjunto de restrições obtém-se que o fim de $\mathbf{a}_1$ deve acontecer antes do fim de $\mathbf{a}_2$, o fim de $\mathbf{a}_2$ deve acontecer antes do fim de $\mathbf{a}_3$ e o fim de $\mathbf{a}_3$ deve acontecer antes do fim de $\mathbf{a}_1$, ou seja $(\mathbf{a}_{1f} < \mathbf{a}_{2f})$ e $(\mathbf{a}_{2f} < \mathbf{a}_{3f})$ e $(\mathbf{a}_{3f} < \mathbf{a}_{1f})$. Por transitividade: $(\mathbf{a}_{1f} < \mathbf{a}_{3f})$ e $(\mathbf{a}_{3f} < \mathbf{a}_{1f})$. Como não existem valores no conjunto dos reais $\Re$ que satisfaçam a este conjunto de inequações, **E3** é inconsistente.

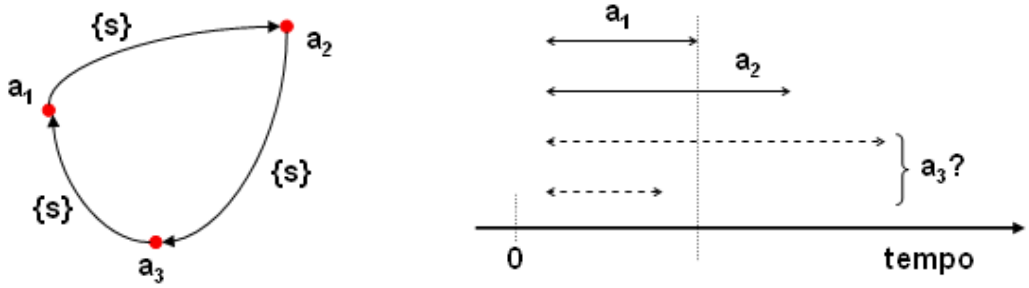


$F3(a_i, a_j)$	Restrições Temporais
$Y_{1,2} = \{s\}$	$a_{1i} = a_{2i}$ e $a_{1f} < a_{2f}$
$Y_{2,3} = \{s\}$	$a_{2i} = a_{3i}$ e $a_{2f} < a_{3f}$
$Y_{3,1} = \{s\}$	$a_{3i} = a_{1i}$ e $a_{3f} < a_{1f}$

(a) Representação Gráfica de **E3** **(b)** Restrições temporais derivadas de **E3**

Figura 3 - 6

Para facilitar a identificação de inconsistências pode-se usar um esquema gráfico. Uma possibilidade é ordenar os intervalos de tempo associados às atividades sobre o eixo do tempo. Cada intervalo é representado por um segmento de reta que é posicionado no eixo do tempo, segundo suas relações temporais, com os demais intervalos. A expressão é consistente se for possível posicionar todos os segmentos de reta. Este processo é denominado **esquema gráfico de consistência** ou **layout de consistência**. A Figura 3 - 7 ilustra este processo para a expressão **E3**.



(a) Representação Gráfica de **E3** **(b)** Layout de consistência de **E3**

Figura 3 - 7

A relação $\mathbf{s(a_1, a_2)}$ determina que o segmento de $\mathbf{a_1}$ deve ser menor que o de $\mathbf{a_2}$ e o extremo inferior de cada intervalo deve coincidir; $\mathbf{s(a_2, a_3)}$ determina que o segmento de $\mathbf{a_2}$ deve ser menor que o de $\mathbf{a_3}$ e o extremo inferior de cada intervalo deve coincidir; $\mathbf{s(a_3, a_1)}$ determina que o segmento de $\mathbf{a_3}$ deve ser menor que o de $\mathbf{a_1}$ e o extremo inferior de cada intervalo deve coincidir, mas o segmento de $\mathbf{a_1}$ é menor que o segmento de $\mathbf{a_3}$. Portanto existe uma inconsistência.

O layout de consistência de uma expressão consistente é único se e somente se as dependências temporais são obtidas a partir de intervalos de tempo fixos. Isto é, após especificar-se os instantes de início e final de cada atividade (nível **N1**) obtém-se de forma única a relação de dependência entre as atividades. Esta afirmação comprova-se porque as relações de dependência satisfazem as propriedades de mútua exclusão e mútua exaustão (Seção 3.1.2).

Lema 1:

Uma expressão linear é consistente

Demonstração

Uma expressão linear formada por duas atividades $\mathbf{a_1}$ e $\mathbf{a_2}$ é consistente conforme estabelecem a Definição 6 e Definição 7.

Suponha-se, por hipótese de indução, que a expressão linear $\mathbf{(E, G)}$, onde $\mathbf{E = (A, R, F)}$, formada por $\mathbf{n}$ atividades $\mathbf{a_1, a_2, \cdots, a_{n-1}, a_n}$ seja consistente.

Verificando-se agora a consistência da expressão linear formada por $\mathbf{n+1}$ atividades tem-se que a atividade $\mathbf{a_{n+1}}$ pode relacionar-se com $\mathbf{a_1}$ ou com $\mathbf{a_n}$, considere-se a relação $\mathbf{r_n(a_n, a_{n+1})}$.

As restrições temporais derivadas de $\mathbf{r_n}$ traduzem o posicionamento do intervalo de tempo associado a $\mathbf{a_{n+1}}$ com relação ao intervalo de tempo associado a $\mathbf{a_n}$ ou vice versa. O posicionamento do intervalo de $\mathbf{a_{n+1}}$ é definido apenas em função do posicionamento do intervalo de $\mathbf{a_n}$ isto é, não depende das restrições temporais derivadas das relações $\mathbf{r_i}$, $\mathbf{i \neq n}$. Logo o intervalo de tempo associado a $\mathbf{a_{n+1}}$ sempre pode ser posicionado sem gerar inconsistências. Portanto a expressão linear formada por $\mathbf{n+1}$ atividades é consistente e fica assim demonstrado o Lema 1 ■.

Teorema 1:

Uma expressão acíclica é consistente.

Demonstração

Suponha-se por absurdo que uma expressão acíclica ε seja inconsistente. Isto significa que existe um conjunto de relações e atividades cujas restrições temporais geram uma inconsistência da forma $(1), (2)$ ou (3):

$$(1)\ x_i < x_j \text{ e } x_i > x_j;$$
$$(2)\ x_i = x_j \text{ e } x_i < x_j;$$
$$(3)\ x_i = x_j \text{ e } x_i > x_j.$$

sendo x_i e x_j variáveis que assumem os valores de tempos de início e/ou fim das atividades.

Sejam **a** e **b** atividades desse conjunto e considere-se:

x_i a variável que se refere ao tempo de início ou fim de **a**, e
x_j a variável que se refere ao tempo de início ou fim de **b**.

Existe um caminho **c** entre **a** e **b**, pois é conexa. As atividades e relações que definem **c** determinam uma expressão linear e, portanto consistente. Assim apenas uma das restrições temporais dos pares especificados em $(1), (2)$ e (3) pode ser considerada.

As restrições temporais que não foram consideradas em cada par também são equações e/ou inequações envolvendo x_i e x_j então existe outro caminho **c'** entre **a** e **b**. Isto determina um ciclo o que é uma contradição, pois a expressão é acíclica. Portanto absurdo é admitir que ε é inconsistente ■.

A inexistência de ciclos em uma expressão é uma condição suficiente para sua consistência, no entanto não é necessária. Uma expressão cíclica pode ser consistente, por exemplo a expressão $\mathbf{E4(A, R, F4)}$, onde

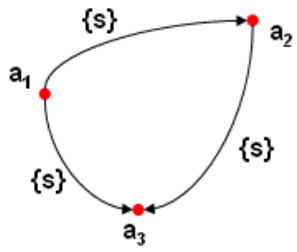
$\mathbf{A} = \{a_1, a_2, a_3\}$, $\mathbf{R} = \{(a_1, a_2), (a_2, a_3), (a_1, a_3)\}$ e

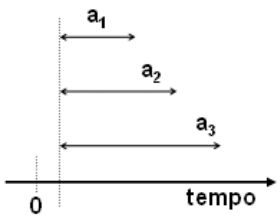
F4 a função rotuladora de arestas definida por:

$$Y_{1,2} = \{s\},\ Y_{2,3} = \{s\} \text{ e } Y_{1,3} = \{s\}$$

é consistente conforme comprova a Figura 3-8.

A expressão **E4** também evidencia uma questão interessante: a ocorrência de redundância na especificação dos comportamentos temporais, isto é, dada uma expressão existem relações que podem ser eliminadas? Por exemplo, a expressão **E4** é uma *expressão redundante* porque a relação $s(a_1, a_3)$ é uma conseqüência das relações $s(a_1, a_2)$ e $s(a_2, a_3)$ e pode ser eliminada sem perda de informação dos comportamentos temporais especificados.



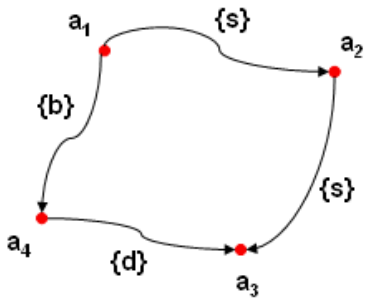


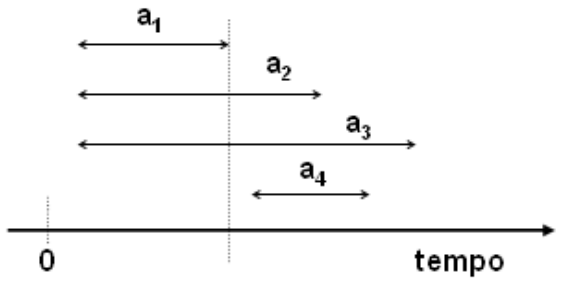
$F4(a_i, a_j)$	Restrições Temp.
$Y_{1,2} = \{s\}$	$a_{1i} = a_{2i}$ e $a_{1f} < a_{2f}$
$Y_{2,3} = \{s\}$	$a_{2i} = a_{3i}$ e $a_{2f} < a_{3f}$
$Y_{1,3} = \{s\}$	$a_{1i} = a_{3i}$ e $a_{1f} < a_{3f}$

(a) Rep. Graf. de **E4** **(b)** Layout de **E4** **(c)** Restrições temporais de **E4**

Figura 3 - 8

No entanto nem toda expressão cíclica consistente é redundante, por exemplo $E5(A, R, F5)$, onde: $A = \{a_1, a_2, a_3, a_4\}$, $R = \{(a_1,a_2),(a_1,a_4),(a_2,a_3),(a_4,a_3)\}$ e **F5** a função rotuladora de arestas por: $Y_{1,2} = \{s\}$, $Y_{1,4} = \{b\}$, $Y_{2,3} = \{s\}$ e $Y_{4,3} = \{d\}$ é cíclica, consistente e não redundante (Figura 3 - 9).





(a) Representação Gráfica de **E5** **(b)** Layout de consistência de **E5**

Figura 3 - 9

3.3. A Construção do Mecanismo de Coordenação

Até este ponto tem-se a especificação dos comportamentos temporais por meio de uma expressão ε da forma como foi estabelecida na Seção 3.2.2. É esta expressão que o nível de abstração **N2** (Figura 3 - 3) recebe e constrói a partir dela o Mecanismo de Coordenação - **MC**.

Nesta seção mostra-se como construir este Mecanismo de Coordenação **MC** para uma expressão consistente. A abordagem de construção compõe-se de dois processos: o de identificação e o de modelagem das Condições Globais **CGs**

(Definição 12). O processo de identificação classifica todas as restrições temporais que diretamente e indiretamente impõem condições para o início e/ou fim de uma atividade $\mathbf{a_i} \in \mathbf{A}$ (**A** é o conjunto de atividades de ε). O processo de modelagem traduz para uma Rede de Petri Colorida estas condições.

Os processos de identificação e modelagem são efetuados em paralelo e por partes. Começa obtendo-se uma partição do conjunto **A**. Para cada elemento pertencente a um subconjunto de **A** procede-se a identificação e a modelagem das restrições temporais obtendo-se o mecanismo de coordenação desse elemento. Ordenadamente, estes mecanismos parciais são conectados obtendo-se ao final o Mecanismo de Coordenação Global.

Estes passos definem o Algoritmo 1 apresentado a seguir.

Algoritmo 1: Identificação e modelagem das **CGs**
Dada uma expressão ε
Obter uma partição do conjunto de atividades de ε
Para cada subconjunto da partição de **A** faça
Para cada elemento do subconjunto selecionado faça
identificar e modelar a lista de restrições temporais obtendo seu mecanismo de coordenação ;
conectar o mecanismo de coordenação anterior com os mecanismos de coordenação apropriados
Fim para
Fim para
Fim algoritmo

Algoritmo 1: Identificação e modelagem das Condições Globais (CGs) - versão1

Na seqüência as Seções 3.3.1 e 3.3.2 introduzem os conceitos e resultados da metodologia **GR** que fundamentam o processo de identificação. A ordem de apresentação segue fluxo do Algoritmo 1. A Seção 3.3.3, de forma análoga, apresenta a fundamentação teórica para o processo de modelagem.

3.3.1 Identificação das Condições Globais - CGs

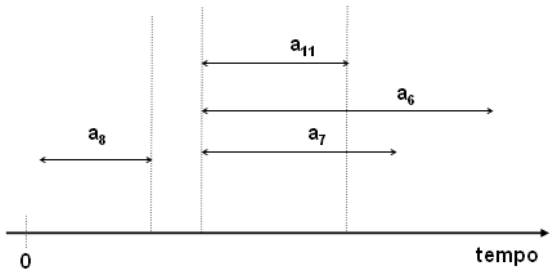
Uma atividade $\mathbf{a_i}$ pode sofrer dois tipos de restrições temporais: as restrições temporais diretas (ou restrições diretas) e as restrições temporais indiretas (ou restrições indiretas). As restrições diretas são derivadas das relações envolvendo diretamente $\mathbf{a_i}$ com outras atividades. As restrições indiretas são inferidas de relações que não envolvem diretamente $\mathbf{a_i}$, mas atividades relacionadas a ela e que impõem condições sobre $\mathbf{a_i}$.

Definição 12: condições globais - CGs

As Condições Globais de uma atividade $\mathbf{a_i}$ são a união de seu conjunto de restrições diretas e de seu conjunto de restrições indiretas.

Por exemplo, analisando-se as condições globais impostas à atividade $\mathbf{a_{11}}$ da expressão $\mathbf{(E2, G2)}$ (Figura 3 - 5) considerando apenas o caminho **c** entre $\mathbf{a_{11}}$ e $\mathbf{a_8}$ constrói-se a tabela apresentada na Figura 3 - 10(a) que lista as restrições temporais derivadas das relações entre as atividades que determinam **c**; elas também podem ser visualizadas no layout de consistência de **c** apresentado na Figura 3 - 10(b).

$\mathbf{F2(a_i, a_j)}$	**Restrições Temporais**
$\mathbf{Y_{11,6} = \{s\}}$	$\mathbf{a_{11i} = a_{6i}}$ e $\mathbf{a_{11f} < a_{6f}}$
$\mathbf{Y_{7,6} = \{s\}}$	$\mathbf{a_{7i} = a_{6i}}$ e $\mathbf{a_{7f} < a_{6f}}$
$\mathbf{Y_{8,7} = \{b\}}$	$\mathbf{a_{8f} < a_{7i}}$



(a) Restrições derivadas do caminho **c** entre $\mathbf{a_{11}}$ e $\mathbf{a_8}$ da expressão $\mathbf{(E2, G2)}$ **(b)** Layout de consistência do caminho **c**

Figura 3 - 10

As restrições diretas de $\mathbf{a_{11}}$, derivadas de $\mathbf{Y_{11,6} = \{s\}}$, determinam que seu início deve ser simultâneo ao início de $\mathbf{a_6}$, $\mathbf{a_{11i} = a_{6i}}$, e que deve ser finalizada antes de $\mathbf{a_6}$, $\mathbf{a_{11f} < a_{6f}}$. Das restrições indiretas de $\mathbf{a_{11}}$, derivadas de $\mathbf{Y_{7,6} = \{s\}}$ e $\mathbf{Y_{8,7} = \{b\}}$, obtém-se que seu início deve ser simultâneo a $\mathbf{a_7}$ e só depois do fim de $\mathbf{a_8}$, conforme análise a seguir (neste exemplo o fim de $\mathbf{a_7}$ e $\mathbf{a_{11}}$ não estão relacionados):

$\mathbf{(a_{11i} = a_{6i})}$ e $\mathbf{(a_{7i} = a_{6i})}$, por transitividade $\mathbf{(a_{11i} = a_{7i})}$;

$\mathbf{(a_{11i} = a_{7i})}$ e $\mathbf{(a_{8f} < a_{7i})} \Rightarrow \mathbf{(a_{8f} < a_{11i})}$.

Este exemplo evidencia a necessidade de uma análise envolvendo todas as relações de uma expressão na determinação das **CGs** de uma atividade $\mathbf{a_i}$. Se esta lista for incompleta não se pode garantir o cumprimento dos comportamentos temporais especificados. No exemplo anterior, $\mathbf{a_{11}}$ pode ocorrer somente após $\mathbf{a_8}$, caso contrário não se pode garantir a relação temporal $\mathbf{b(a_8, a_7)}$.

As **CGs** de uma atividade $\mathbf{a_i}$ podem ser identificadas analisando-se exaustivamente todos os caminhos entre $\mathbf{a_i}$ e $\mathbf{a_j}$, $\forall\, \mathbf{j \neq i}$. No entanto resolver este

problema adotando uma abordagem exaustiva não é uma alternativa tratável no caso geral. Este algoritmo "força bruta" tem complexidade exponencial no número de atividades para o pior caso como é o caso das expressões lineares (Definição 10), cujos rótulos de todas as arestas têm cardinalidade 7, i.e:

$$\mathbf{F}(\mathbf{a_i}, \mathbf{a_j}) = \{\mathbf{e, s, d, f, o, m, b}\},\ \forall(\mathbf{a_i}, \mathbf{a_j}) \in \mathbf{R}$$

Proposição 1:

O algoritmo "força bruta" para determinar as Condições Globais **CGs** de uma expressão tem complexidade exponencial, em relação ao número de atividades, para o pior caso.

Demonstração

Dada uma expressão linear, cuja função rotuladora de aresta $\mathbf{F}$ é tal que

$$\mathbf{F}(\mathbf{a_i}, \mathbf{a_j}) = \{\mathbf{e, s, d, f, o, m, b}\},\ \forall(\mathbf{a_i}, \mathbf{a_j}) \in \mathbf{R}.$$

A função $\mathbf{F}$ associa a cada aresta um rótulo com 7 primitivas. Isto determina $\mathbf{7^{n-1}}$, sendo $\mathbf{n}$ o número de atividades da expressão, possíveis alternativas. Cada alternativa é determinada selecionando-se uma das 7 primitivas do rótulo de cada aresta. Considere-se uma destas alternativas.

Para cada $\mathbf{a_i} \in \mathbf{A}$, $\mathbf{i = 1,2,\cdots,n}$, existem :

1. um caminho $\mathbf{c = a_1, \cdots, a_i}$ de comprimento (número de arestas) $|\mathbf{c}| = \mathbf{p}$ e um caminho $\mathbf{c' = a_i, \cdots, a_n}$ de comprimento $|\mathbf{c'}| = \mathbf{q}$ tal que $\mathbf{p + q = n - 1}$;
2. **n-1** caminhos $\mathbf{c_k = a_i, \cdots, a_k}$, onde $\mathbf{k = 1,2,\cdots,n}$ e $\mathbf{k \neq i}$, de comprimentos $\underbrace{\mathbf{1,2,\cdots,p,1,2,\cdots,q}}_{\mathbf{n-1}\ \text{valores}}$.

Considere-se o comprimento (número de arestas) de um caminho como sendo o seu custo de análise. Assim o custo para analisar as **CGs** de uma atividade $\mathbf{a_i}$ é a somatória dos comprimentos dos seus $\mathbf{n-1}$ caminhos. De 1. e 2. tem-se que:

$$\mathbf{custo}(\mathbf{a_i}) = \sum_{\mathbf{j=1}}^{\mathbf{p}} \mathbf{j} + \sum_{\mathbf{j=1}}^{\mathbf{q}} \mathbf{j} = \frac{\mathbf{p(1+p)}}{\mathbf{2}} + \frac{\mathbf{q(1+q)}}{\mathbf{2}} = \frac{\mathbf{(p+q)^2 - 2pq + p + q}}{\mathbf{2}} =$$

$$= \frac{\mathbf{(n-1)^2 + (n-1) - 2pq}}{\mathbf{2}}.$$

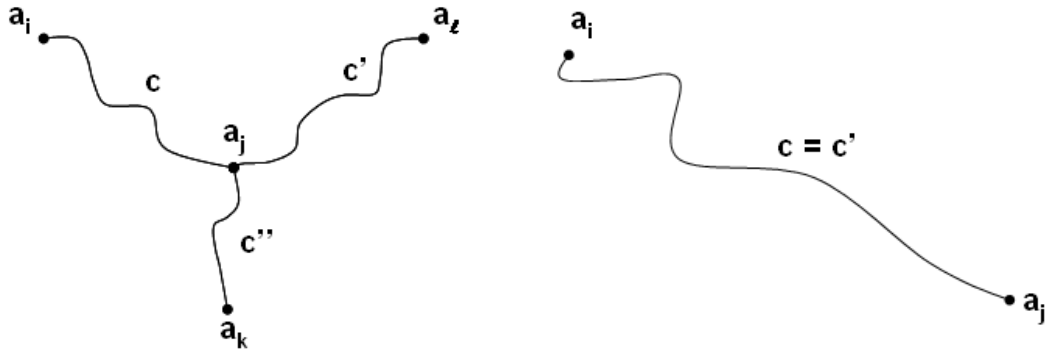
Logo a complexidade para avaliar as **n** atividades da expressão, para uma alternativa, é da ordem de $\mathbf{O(n^3)}$. Portanto o algoritmo tem complexidade exponencial, pois existem $\mathbf{7^{n-1}}$ alternativas para serem analisadas .

A complexidade exponencial do algoritmo "força bruta" dá-se em função dos seguintes fatos:

(a) Confluência de caminhos: dois caminhos $\mathbf{c = a_i, \cdots, a_j, \cdots, a_k}$ e $\mathbf{c' = a_\ell, \cdots, a_j, \cdots, a_k}$ apresentam, a partir de uma dada atividade $\mathbf{a_j}$ um sub-caminho **c''** comum a **c** e **c'**. Este trecho comum aos dois caminhos não é percebido pelo algoritmo da força bruta e é analisado tanto em **c** quanto em **c'** (Figura 3-11(a)).

(b) Coincidência de caminhos: os caminhos $\mathbf{c = a_i, \cdots, a_j}$ e $\mathbf{c' = a_j, \cdots, a_i}$ são formados pelo mesmo conjunto de atividades e conseqüentemente pelas mesmas relações temporais. A diferença entre **c** e **c'** é apenas a ordem das seqüências. Do ponto de vista da identificação das **CGs** é suficiente analisar **c** ou **c'**. O algoritmo "força bruta" faz esta análise tanto para **c** quanto para **c'** (Figura 3-11(b)).

Estes problemas combinatórios podem ser contornados adotando-se estratégias que reconheçam as situações descritas acima. Situações que foram provocadas essencialmente porque o algoritmo trata cegamente todo um caminho **c** para depois analisar um outro caminho **c'**.



(a) a seqüência $\mathbf{c'' = a_j, \cdots, a_k}$, comum aos caminhos $\mathbf{c = a_i, \cdots, a_j, \cdots, a_k}$ e $\mathbf{c' = a_\ell, \cdots, a_j, \cdots, a_k}$, é avaliada várias vezes na identificação das **CGs** de $\mathbf{a_i}$.

(b) o caminho $\mathbf{c = a_i, \cdots, a_j}$ é percorrido para analisar as **CGs** de $\mathbf{a_i}$ e também é percorrido, no sentido contrário $\mathbf{c' = a_j, \cdots, a_i}$, quando esta sendo analisado as **CGs** de $\mathbf{a_j}$.

Figura 3-11: Configurações não percebidas pelo algoritmo da força bruta.

Considerando tal fato, na busca de um algoritmo de menor complexidade, conclui-se que uma boa estratégia é efetuar análises por partes e trabalhar simultaneamente com o maior número possível de caminhos.

3.3.2 Algoritmo proposto para identificação das Condições Globais (CGs)

O Algoritmo 1 proposto para determinar as **CGs** de uma expressão supera os problemas expostos na seção anterior visitando todos os caminhos uma única vez e percorrendo-os por partes.

A estratégia adotada consiste em gerar, a partir da expressão original ε, uma seqüência de sub-expressões $\varepsilon_0, \varepsilon_1, \cdots, \varepsilon_m$; para cada ε_i seleciona-se um conjunto apropriado de atividades $\mathbf{a}_i$ e determina-se a lista de restrições temporais diretas de cada $\mathbf{a}_i$. Estas listas são devidamente conectadas às listas das atividades selecionadas em ε_{i+1}. As restrições temporais indiretas das atividades vão sendo determinadas nesta operação de conexão entre as listas de restrições temporais diretas.

Dessa forma o Algoritmo 1 começa processando as atividades mais externas e vai caminhando para as atividades mais internas até atingir o conjunto de atividades derivado expressão ε_m. O conjunto derivado de ε_m é denominado centro da expressão ε que corresponde ao centro de um grafo, conforme a definição apresentada a seguir.

> "O centro de um grafo é o subconjunto dos vértices de excentricidade mínima". Sendo a excentricidade de um vértice a distância máxima entre ele e qualquer outro vértice do grafo [**Szwarcfiter 86**].

O Algoritmo 1 processa a cada iteração uma sub-expressão ε_i, $\mathbf{i = 1,2,\cdots,m}$. O processo de identificação das **CGs** chega ao fim quando o algoritmo alcança o centro da expressão. Ao fim do processo todas as $\mathbf{n-1}$ relações, onde **n** é a quantidade de atividades, terão sido analisadas uma única vez em **m** iterações. Apresentam-se a seguir os resultados que fundamentam os três passos principais de cada iteração.

Resultados que fundamentam o Algoritmo 1

Passo 1 – Selecionar um conjunto $\mathbf{I}_k$ de atividades

A seleção do conjunto $\mathbf{I}_k$ da iteração atual é feita a partir da definição de sub-expressões de ε (Definição 13) e da partição do conjunto **A** (Definição 14).

Embora o processo de identificação possa ser iniciado por qualquer conjunto de atividades é interessante definir uma ordem de seleção das atividades. Isto traz vantagens para o processo de identificação das **CGs**, pois pode-se explorar a priori o fato de uma atividade $\mathbf{a_i}$ ser folha (atividade com apenas uma relação de dependência $\partial(\mathbf{a_i})=\mathbf{1}$) ou interna (atividade com mais de uma relação de dependência $\partial(\mathbf{a_i})>\mathbf{1}$) de forma a simplificar também o processo de modelagem (Seção 3.3.3).

A fim de estabelecer esta ordem, dada uma expressão $\boldsymbol{\varepsilon}$, obtém-se uma partição do conjunto de atividades **A**, denotada por **P(A)**. Os subconjuntos que determinam **P(A)** são determinados através de uma seqüência de expressões $\{\boldsymbol{\varepsilon_i}\}$, $\mathbf{i=0,1,2,\cdots,m}$, obtida a partir da expressão original $\boldsymbol{\varepsilon}$. Cada expressão $\{\boldsymbol{\varepsilon_i}\}$ dá origem a um elemento de **P(A)**. A Definição 13, apresentada a seguir, estabelece a lei de formação da seqüência $\{\boldsymbol{\varepsilon_i}\}$. A Definição 14 diz como obter os elementos de **P(A)** conhecendo-se seqüência $\{\boldsymbol{\varepsilon_i}\}$.

Definição 13: seqüência $\{\boldsymbol{\varepsilon_i}\}$

$\{\boldsymbol{\varepsilon_i}\}$ é uma seqüência de sub-expressões obtida de uma expressão $\boldsymbol{\varepsilon}$ da metodologia **GR** obedecendo a seguinte lei de formação:

$\boldsymbol{\varepsilon_i}$	**Lei de formação**
$\boldsymbol{\varepsilon_0}$	é a expressão original $\boldsymbol{\varepsilon}$;
$\boldsymbol{\varepsilon_1}$	Expressão derivada de $\boldsymbol{\varepsilon_0}$ eliminando as atividades de grau um de $\boldsymbol{\varepsilon_0}$;
$\boldsymbol{\varepsilon_2}$	Expressão derivada de $\boldsymbol{\varepsilon_1}$ eliminando as atividades de grau um de $\boldsymbol{\varepsilon_1}$;
$\vdots$	$\vdots$
$\boldsymbol{\varepsilon_m}$	Expressão derivada de $\boldsymbol{\varepsilon_{m-1}}$ eliminando as atividades de grau um de $\boldsymbol{\varepsilon_{m-1}}$.

Conclui-se da Definição 13 que a seqüência $\{\boldsymbol{\varepsilon_i}\}$ tem tamanho máximo **m+1** menor ou igual a metade do número de atividades **n** envolvidas em uma expressão $\boldsymbol{\varepsilon}$, isto é, $\mathbf{m} \le \dfrac{\mathbf{n-2}}{\mathbf{2}}$. O maior valor de **m** acontece para as expressões lineares (Definição 10).

Definição 14: partição do conjunto de atividades A

Uma partição **P** do conjunto de atividades **A** de uma expressão $\boldsymbol{\varepsilon}$ é dada por $\mathbf{P(A)}=\{\mathbf{I_0,I_1,\cdots,I_m}\}$, onde $\mathbf{m} \le \dfrac{\mathbf{n}}{\mathbf{2}}$, sendo **n** o número de atividades e

$$\mathbf{I}_0 = \{\mathbf{a}_i \in \varepsilon_0 \,|\, \partial(\mathbf{a}_i) = 1\}$$
$$\mathbf{I}_1 = \{\mathbf{a}_i \in \varepsilon_1 \,|\, \partial(\mathbf{a}_i) = 1\}$$
$$\mathbf{I}_2 = \{\mathbf{a}_i \in \varepsilon_2 \,|\, \partial(\mathbf{a}_i) = 1\}$$
$$\vdots$$
$$\mathbf{I}_m = \{\mathbf{a}_i \in \varepsilon_m \,|\, \partial(\mathbf{a}_i) = 1\}$$

A Figura 3 – 12 a seguir ilustra a representação gráfica de uma expressão e a partição do seu conjunto de atividades.

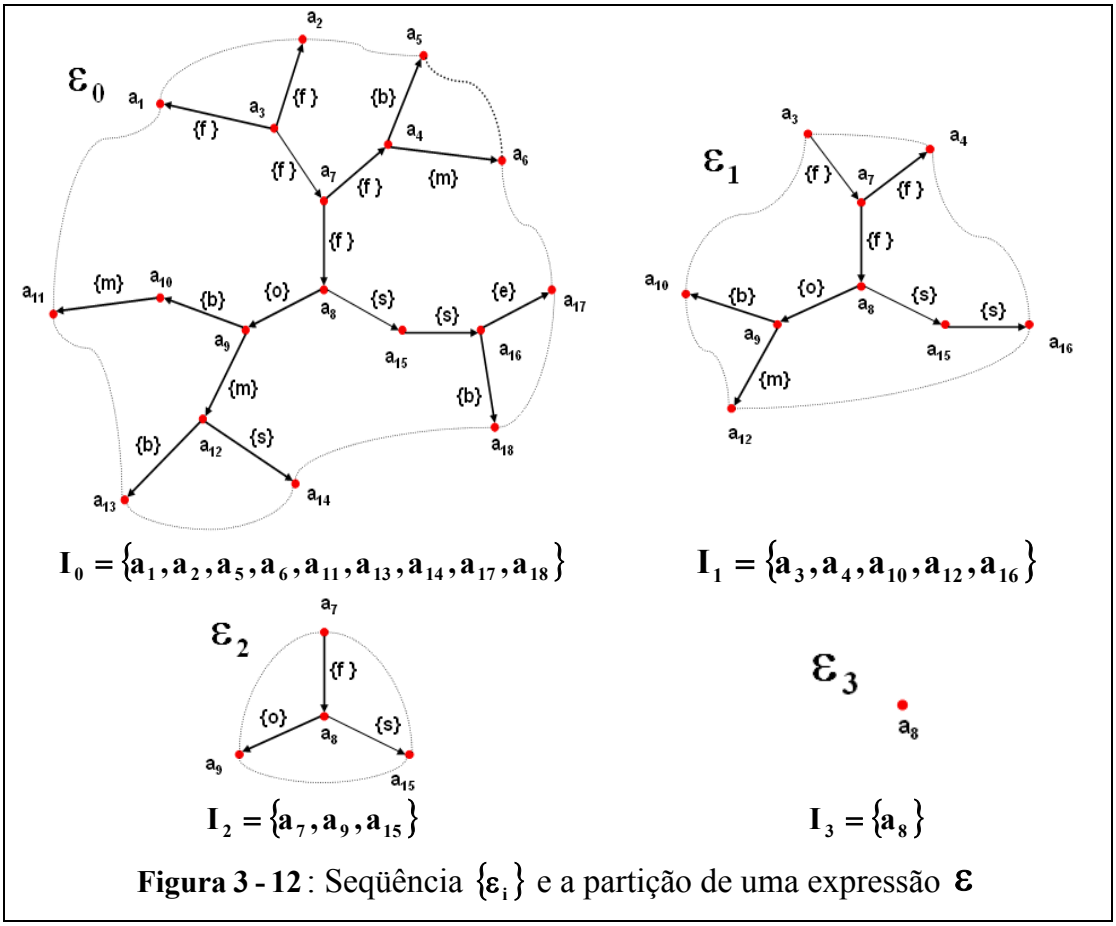


Figura 3 - 12: Seqüência $\{\varepsilon_i\}$ e a partição de uma expressão ε

A finalidade da seqüência $\{\varepsilon_i\}$ é a de formalizar o critério de partição de **A**. De fato não é necessário, nem conveniente criar efetivamente $\{\varepsilon_i\}$. É suficiente computar uma lista contendo o grau de todas as atividades; assim $\mathbf{I}_0$ é determinado selecionando as atividades $\mathbf{a}_i$ da lista tal que $\partial(\mathbf{a}_i) = 1$. Para determinar $\mathbf{I}_1$ é necessário atualizar a lista. Para cada atividade $\mathbf{a}_i$ de $\mathbf{I}_0$: (1) retirar $\mathbf{a}_i$ da lista; (2) subtrair uma unidade do grau da atividade $\mathbf{a}_j$ relacionada com $\mathbf{a}_i$. Então, $\mathbf{I}_1$ é

determinado por todas as atividades $\mathbf{a_j}$, $\mathbf{j \neq i}$ tal que $\partial(\mathbf{a_j}) = \mathbf{1}$. Este procedimento é finalizado quanto a lista contiver apenas uma ou duas atividades, isto é, o centro da expressão, que corresponde ao conjunto $\mathbf{I_m}$.

Os elementos de **P(A)** podem ser vistos como "curvas de nível" de ε, $\mathbf{I_0}$ correspondendo a curva de menor nível e $\mathbf{I_m}$ a curva de maior nível. Tomando como referência a expressão original ε, $\mathbf{I_0}$ é de fato o único conjunto formado por atividades $\mathbf{a_i}$ tal que $\partial(\mathbf{a_i}) = \mathbf{1}$. Uma propriedade deste critério de partição é que as atividades que estão sobre a mesma curva de nível, a menos da curva $\mathbf{I_m}$, não estabelecem relacionamentos entre si.

Uma vez estabelecido o critério de partição, convenciona-se, sem perda de generalidade, uma ordem de seleção de fora para dentro, isto é, das atividades mais externas, conjunto $\mathbf{I_1}$, para as mais internas, conjuntos $\mathbf{I_2, I_3, \cdots, I_m}$. Uma vantagem desta ordem de seleção é começar por $\mathbf{I_1}$ e não por $\mathbf{I_0}$. As restrições temporais de $\mathbf{I_0}$ podem ser determinadas quando forem processadas as atividades de $\mathbf{I_1}$.

A curva do nível **m** sempre terá uma ou duas atividades. Este fato se verifica porque a expressão original, retirado o sentido das arestas, é um grafo acíclico e conexo e o centro de um grafo com estas características tem um ou dois vértices [Szwarcfiter 86].

Passo 2 – Determinar as Condições Globais para cada $\mathbf{a_i \in I_k}$

Selecionado o conjunto de atividades $\mathbf{a_i \in I_k}$, o próximo passo é determinar para cada $\mathbf{a_i}$ sua lista de restrições diretas. Para isto é necessário conhecer o conjunto de atividades $\mathbf{a_j}$ relacionadas a ela. A Proposição 2 mostra que estas atividades pertencem às curvas $\mathbf{I_{k-1}}$ e $\mathbf{I_{k+1}}$, podendo existir atividades em $\mathbf{I_0}$. A Proposição 2 permite concluir que $\mathbf{a_i}$ não se relaciona com atividades que pertençam a sua curva de nível.

Outro resultado usado na construção do algoritmo é estabelecido pela Proposição 3 que garante que uma atividade $\mathbf{a_i}$ do nível $\mathbf{I_k}$ relaciona-se com apenas uma atividade do nível $\mathbf{I_{k+1}}$.

Proposição 2:

> Seja ε uma expressão da metodologia **GR**, $\mathbf{I_0, I_{k-1}, I_k, I_{k+1} \in P(A)}$ e $\mathbf{a_i, a_j \in A}$. Se $\mathbf{a_j}$ é uma atividade relacionada a $\mathbf{a_i \in I_k, k \neq 0}$ então $\mathbf{a_j \in I_0}$ ou $\mathbf{a_j \in I_{k-1}}$ ou $\mathbf{a_j \in I_{k+1}}$.

Demonstração

Considere-se $\mathbf{a_j \in I_q}$, para $\mathbf{q \in \{0,1,2,\cdots,m\}}$. Por hipótese tem-se $\mathbf{r(a_i, a_j)}$ para $\mathbf{r \in D}$. Existem duas alternativas, $\mathbf{q = 0}$ ou $\mathbf{q \neq 0}$. Para $\mathbf{q = 0}$ tem-se o caso trivial.

Suponha-se $\mathbf{q \neq 0}$, por hipótese $\mathbf{a_i \in I_k}$ e existe um único caminho **c** entre $\mathbf{a_i}$ e $\mathbf{a_j}$, pois $\boldsymbol{\varepsilon}$ é conexa e acíclica. Como existe a relação $\mathbf{r(a_i, a_j)}$ o comprimento de **c** é necessariamente 1**,** $\mathbf{|c| = 1}$. Logo dizer $\mathbf{a_j \in I_q}$ significa dizer $\mathbf{q = 0}$ ou $\mathbf{q = k - 1}$ ou $\mathbf{q = k + 1}$□.

Proposição 3:

> Seja $\boldsymbol{\varepsilon}$ uma expressão da metodologia **GR,** $\mathbf{I_k, I_{k+1} \in P(A)}$. Se $\mathbf{r(a_i, a_j)}$, $\mathbf{a_i \in I_k}$ e $\mathbf{a_j \in I_{k+1}}$ então $\mathbf{a_j}$ é a única atividade do nível $\mathbf{I_{k+1}}$ relacionada à $\mathbf{a_i}$.

Demonstração

Se $\mathbf{a_i \in I_k}$ então existe uma expressão $\boldsymbol{\varepsilon_k}$, derivada de $\boldsymbol{\varepsilon}$ da forma como estabelece a Definição 13 e a Definição 14, tal que $\mathbf{\partial(a_i) = 1}$, isto é, existe uma única atividade relacionada à $\mathbf{a_i}$. Segue da hipótese que esta atividades é $\mathbf{a_j}$, pois $\mathbf{a_j \in I_{k+1}}$. Fica assim demonstrada a Proposição 3□.

A fim de facilitar a referência ao conjunto de atividades relacionadas à $\mathbf{a_i}$, estabelece-se o termo *estrela* $\mathbf{a_i}$ da forma como determina a Definição 15.

Definição 15: estrela $\mathbf{a_i}$

> Uma estrela de centro $\mathbf{a_i}$, ou simplesmente estrela $\mathbf{a_i}$, é uma sub-expressão determinada por $\mathbf{a_i}$, pelo conjunto de atividades $\mathbf{a_j}$ relacionadas à $\mathbf{a_i}$ e por suas respectivas relações.

O passo 2 do algoritmo inicia analisando as estrelas com centros em $\mathbf{I_1}$, depois com centros em $\mathbf{I_2}$ até finalizar ao atingir a curva $\mathbf{I_m}$.

Analisar uma estrela $\mathbf{a_i}$ significa determinar o conjunto de restrições diretas que a atividade $\mathbf{a_i}$ deve satisfazer, denominado lista de restrições diretas de $\mathbf{a_i}$, isto é $\mathbf{La_i}$. Para isto considere-se **A** e **B** dois conjuntos de restrições temporais e as seguintes convenções:

- $\mathbf{A\ .e.\ B}$ é o conjunto formado por todas as restrições temporais de **A** e de **B**, lê-se **A** conjunção **B**;

- $\mathbf{A}$ **.ou.** $\mathbf{B}$ é o conjunto formado por todas as restrições temporais de **A** ou exclusivamente de **B**, lê-se **A** disjunção **B**.

A Definição 16 estabelece as regras para determinação do conjunto de restrições diretas de uma estrela $\mathbf{a_i}$.

Definição 16: restrições diretas da estrela $\mathbf{a_i}$

As restrições diretas da estrela $\mathbf{a_i}$, isto é $\mathbf{La_i}$ são as restrições temporais derivadas de todas as relações envolvendo $\mathbf{a_i}$ e formadas pela conjunção de $\mathbf{C_1}$, $\mathbf{C_2}$ e $\mathbf{C_3}$, isto é, $\mathbf{La_i = C_1 .e. C_2 .e. C_3}$, onde

(1) $\mathbf{C_1}$ é a conjunção dos conjuntos de restrições derivadas das relações entre $\mathbf{a_i}$ e as atividades $\mathbf{a_j}$ que satisfazem as seguintes propriedades:

i. $\mathbf{a_j \notin G(a_i)}$;

ii. o rótulo da aresta definida por $\mathbf{a_i}$ e $\mathbf{a_j}$ é unitário (possui uma primitiva);

(2) $\mathbf{C_2}$ é a disjunção dos conjuntos de restrições derivadas das relações entre $\mathbf{a_i}$ e as atividades $\mathbf{a_j}$ tal que:

i. $\mathbf{a_j \in G(a_i)}$;

(3) $\mathbf{C_3}$ é a conjunção dos $\mathbf{C_{ij}}$, onde cada $\mathbf{C_{ij}}$ é a disjunção dos conjuntos de restrições derivadas das relações do rótulo da aresta definida por $\mathbf{a_i}$ e $\mathbf{a_j}$ se $\mathbf{a_j}$ satisfaz as propriedades a seguir:

i. $\mathbf{a_j \notin G(a_i)}$,

ii. o rótulo da aresta definida por $\mathbf{a_i}$ e $\mathbf{a_j}$ é não unitário.

Para um exemplo das restrições diretas de uma estrela, considere-se a estrela $\mathbf{a_1}$ da expressão qualificada $\mathbf{(E6, G6)}$, ilustrada na Figura 3-13, onde $\mathbf{E6(A, R, F6)}$ é dada por:

$$\mathbf{A = \{a_1, a_2, a_3, a_4, a_5, a_5\},\ R = \{(a_1, a_2),\ (a_1, a_3),\ (a_1, a_4), (a_1, a_5), (a_1, a_6)\}},$$

F6 a função rotuladora de arestas dada por:

$$\mathbf{Y_{1,2} = \{b\},\ Y_{1,3} = \{e\},\ Y_{1,4} = \{s\},\ Y_{1,5} = \{e, s\},\ Y_{1,6} = \{b\}} \text{ e}$$

G6 a função rotuladora de atividades dada por:

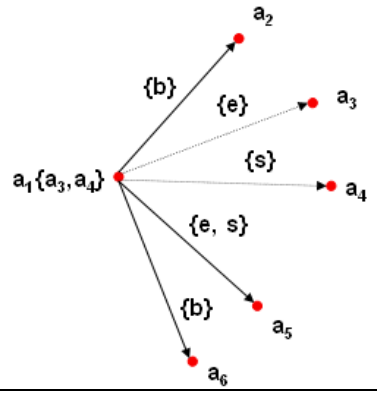
$$\mathbf{X_1 = \{a_3, a_4\}} \text{ e } \mathbf{X_2 = X_3 = X_4 = X_5 = X_6 = \varnothing}.$$

Para determinar os conjuntos $\mathbf{C_1}$, $\mathbf{C_2}$ e $\mathbf{C_3}$ que definem as restrições diretas da estrela $\mathbf{a_1}$ identificam-se, usando a Definição 16, as relações que

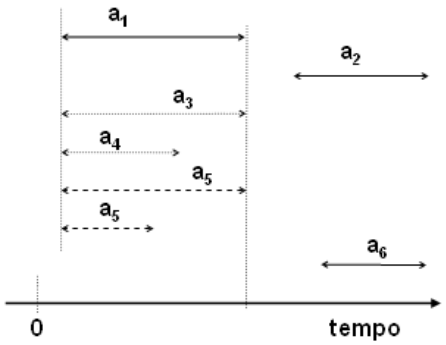
pertencem a cada um deles e da Definição 6 obtém-se as restrições temporais derivada de cada relação. Lista-se a seguir os referidos conjuntos:

$$\mathbf{C_1 = \{Y_{1,2}\ .e.\ Y_{1,6}\} = \{\{b\}.e.\{b\}\} =}$$
$$\mathbf{= \{(a_{1f} < a_{2i}).e.(a_{1f} < a_{6i})\}}$$
$$\mathbf{C_2 = \{Y_{1,3}\ .ou.\ Y_{1,4}\} = \{\{e\}.ou.\{s\}\} =}$$
$$\mathbf{= \{((a_{1i} = a_{3i}).e.(a_{1f} = a_{3f})).ou.((a_{1i} = a_{4i}).e.(a_{1f} < a_{4f}))\}}$$
$$\mathbf{C_3 = C_{15} = \{Y_{1,5}\} = \{\{e\}.ou.\{s\}\} =}$$
$$\mathbf{= \{((a_{1i} = a_{5i}).e.(a_{1f} = a_{5f})).ou.((a_{1i} = a_{5i}).e.(a_{1f} < a_{5f}))\}.}$$

Assim, o conjunto de restrições temporais de $\mathbf{a_1}$ é definido como: $\mathbf{La_1 = C_1\ .e.\ C_2\ .e.\ C_3}$.



(a)



(b) $\mathbf{Y_{1,5}}$ determina duas representações para $\mathbf{a_5}$

Figura 3-13: **(a)** Gráfico de $\mathbf{(E6, G6)}$ **(b)** Layout de consistência de $\mathbf{(E6, G6)}$.

O terceiro passo do algoritmo identifica a dependência entre os centros de estrelas adjacentes (Definição_17) e promove a conexão de suas listas de restrições temporais.

Passo 3 – Conectar as listas de Condições Globais

Se uma estrela $\mathbf{a_i}$ é adjacente à uma estrela $\mathbf{a_j}$, então existe uma única relação $\mathbf{r \in D}$ entre os centros $\mathbf{a_i}$ e $\mathbf{a_j}$, conforme demonstrado na Proposição 3. São as restrições diretas derivadas de **r** que estabelecem as condições para efetuar a conexão entre a lista de restrições diretas de $\mathbf{a_i}$ (isto é, $\mathbf{La_i}$) e a lista de restrições

diretas de $\mathbf{a_j}$ (isto é, $\mathbf{La_j}$). Dessa forma, impõem-se à $\mathbf{a_i}$ e $\mathbf{a_j}$ as restrições temporais derivadas indiretamente de outras relações, isto é, as restrições indiretas.

Definição 17: estrelas adjacentes

Uma estrela $\mathbf{a_i}$ é adjacente a uma estrela $\mathbf{a_j}$ se existe uma relação entre $\mathbf{a_i}$ e $\mathbf{a_j}$, isto é, $\mathbf{F(a_i, a_j) \neq \varnothing}$ ou $\mathbf{F(a_j, a_i) \neq \varnothing}$.

A título de exemplo, considere-se estrela $\mathbf{a_4}$ da a expressão $\mathbf{(E7, G7)}$ construída de $\mathbf{(E6, G6)}$ adicionando-se as atividades $\mathbf{a_7}$ e $\mathbf{a_8}$ (ver Figura 3 - 14):

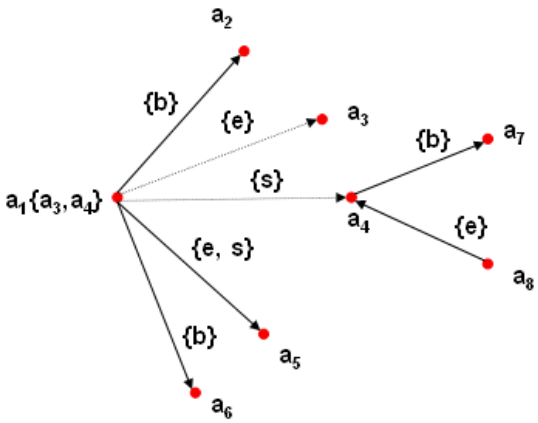


Figura 3 - 14 : Gráfico de $\mathbf{(E7, G7)}$.

$\mathbf{A = \{a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8\}}$,
$\mathbf{R = \{(a_1, a_2), (a_1, a_3), (a_1, a_4), (a_1, a_5), (a_1, a_6), (a_4, a_7), (a_8, a_4)\}}$

F7 a função rotuladora de arestas dada por:

$\mathbf{Y_{1,2} = \{b\}}$, $\mathbf{Y_{1,3} = \{e\}}$, $\mathbf{Y_{1,4} = \{s\}}$, $\mathbf{Y_{1,5} = \{e, s\}}$, $\mathbf{Y_{1,6} = \{b\}}$, $\mathbf{Y_{4,7} = \{b\}}$ e $\mathbf{Y_{8,4} = \{e\}}$

G7 a função rotuladora de atividades dada por:

$$\mathbf{X_1 = \{a_3, a_4\}} \text{ e } \mathbf{X_2 = X_3 = \cdots = X_7 = X_8 = \varnothing}.$$

Aplicando-se a Definição 16 para estrela $\mathbf{a_4}$ obtém-se:

$$\mathbf{C_1 = \{Y_{1,4} .e. Y_{4,7} .e. Y_{8,4}\} = \{\{s\}.e.\{b\}.e.\{e\}\} =}$$
$$\mathbf{= \{((a_{1i} = a_{4i}).e.(a_{1f} = a_{4f})).e.(a_{7f} < a_{4i}).e.((a_{8i} = a_{4i}).e.(a_{8f} = a_{4f}))\},}$$
$$\mathbf{C_2 = C_3 = \varnothing}$$
$$\mathbf{La_4 = \{((a_{1i} = a_{4i}).e.(a_{1f} = a_{4f})).e.(a_{7f} < a_{4i}).e.((a_{8i} = a_{4i}).e.(a_{8f} = a_{4f}))\}.}$$

A conexão entre $\mathbf{La_1}$ e $\mathbf{La_4}$ são as restrições diretas derivadas da relação **starts** entre as estrelas $\mathbf{a_1}$ e $\mathbf{a_4}$: $\left((\mathbf{a_{1i}} = \mathbf{a_{4i}}).\mathbf{e}.(\mathbf{a_{1f}} = \mathbf{a_{4f}})\right)$. Estas restrições determinam que o início de $\mathbf{a_1}$ depende do início de $\mathbf{a_4}$. Verificando-se as restrições indiretas observa-se de $\mathbf{La_4}$ que o início de $\mathbf{a_4}$ depende do início de $\mathbf{a_8}$, logo $\mathbf{a_1}$ depende do início de $\mathbf{a_8}$.

Uma vez identificadas estas restrições, o processo de modelagem, apresentado a seguir, promove a conexão entre o Mecanismos de Coordenação da estrela $\mathbf{a_i}$ isto é, $\mathbf{MCa_i}$ com o Mecanismo de Coordenação da estrela $\mathbf{a_j}$, isto é $\mathbf{MCa_j}$. Efetuando-se esta operação para todas as estrelas obtém-se o mecanismo de coordenação para os comportamentos temporais especificados em uma expressão.

3.3.3. Modelagem das Condições Globais

Apresentam-se nesta seção as operações usadas no processo de modelagem das **CGs** que é efetuado simultaneamente ao processo de identificação já introduzido.

Para a modelagem das **CGs** inicialmente são modeladas as restrições diretas de todas as estrelas $\mathbf{a_i}$ iniciando pela curva de nível $\mathbf{I_1}$ e continuando sucessivamente até a curva de nível $\mathbf{I_m}$.

Para cada uma das estrelas $\mathbf{a_i}$ do nível **k** (passo 1 do Algoritmo 1), o processo inicia modelando as restrições diretas derivadas de cada relação **r**, obtendo-se assim o mecanismo de coordenação para cada **r** de $\mathbf{a_i}$. Estes mecanismos são então conectados gerando o mecanismo da estrela $\mathbf{a_i}$ (passo 2 do Algoritmo 1).

Por fim, efetua-se a conexão dos mecanismos gerados para as estrelas $\mathbf{a_i}$ do nível **k** com os mecanismos gerados para as estrelas $\mathbf{a_j}$ do nível $\mathbf{k-1}$ (passo 3 do Algoritmo 1). Segue-se com este procedimento até a última curva de nível[13], obtendo-se desta forma o Mecanismo de Coordenação **MC** da expressão ε.

Este **MC** de ε obtido é o modelo do coordenador da aplicação a ser implementado no nível de abstração **N3** (nível de execução) oferecendo dois mecanismos de coordenação de atividades:

- **MCL -** Mecanismo de Coordenação Local
- **MCG -** Mecanismo de Coordenação Global.

Em seguida são apresentados os detalhes de obtenção de cada um dos mecanismos de coordenação – local e global – durante o processamento do Algoritmo 1.

---

[13] As restrições diretas das atividades do nível 0 são modeladas com as do nível 1

3.3.3.1. Mecanismo de Coordenação Local MCL

MCLs encapsulam a execução "propriamente dita" das atividades. São ditas locais em contraposição aos **MCGs**, pois são dirigidos a coordenação de uma relação entre duas atividades - $\mathbf{a_j}$ e $\mathbf{a_k}$ - abstraindo-se em termos dos **MCGs**, as relações de $\mathbf{a_j}$ e $\mathbf{a_k}$ com as outras atividades existentes.

O modelo matemático usado para representar a funcionalidade do **MCL** é a Rede de Petri [Murata 89]. O diagrama proposto para coordenar o tempo de execução de uma atividade **x** é ilustrado na Figura 3-15.

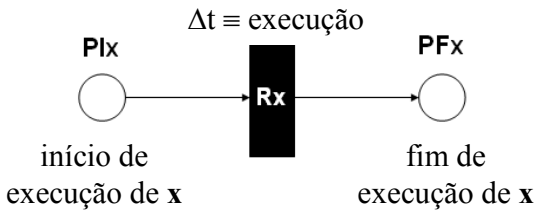


Figura 3-15: **MCL** relativo à atividade **x**.

Quando uma ficha alcança o **MCL** que coordena um par de atividades $\mathbf{a_j}$, $\mathbf{a_k}$ é porque as condições globais impostas para uma ou à ambas foram satisfeitas. Começa então o trabalho do **MCL**, cuja competência é coordenar o tempo gasto para execução das atividades relacionadas.

Uma ficha no lugar **PIx** habilita o início da execução de **x**. A transição **Rx** representa o intervalo de tempo (Δt) despendido na execução de **x**. Para representar o tempo gasto na execução de uma atividade, transição **Rx**, emprega-se o conceito da transição com reserva de fichas [Ramamoorthy 80]. Neste tipo de transição o disparo acontece em dois momentos. Primeiro as fichas são retiradas do lugar de entrada **PIx** quando a transição está habilitada; segundo, as fichas são enviadas para o lugar de saída **PFx** após despender um determinado tempo. O disparo desta transição deve corresponder ao início de **x** no nível de abstração **N3** (nível de execução).

O fim de execução de **x** é determinado por uma ficha no lugar **PFx**. A Figura 3-16 apresenta o mecanismo de coordenação local **MCL** para as relações definidas em **D** (Definição 6).

Nas próximas seções os **MCLs** são representados graficamente por "caixas pretas" que devem ser substituídas pelos mecanismos correspondentes. Por exemplo, na Figura 3-19 (a) e Figura 3-21 (a) as "caixas pretas" rotuladas por:

$\mathbf{b(a_j, a_k)}$: $\mathbf{a_j}$ before $\mathbf{a_k}$ e

$\mathbf{s(a_j, a_k)}$: $\mathbf{a_j}$ starts $\mathbf{a_k}$

devem ser substituídas pelos **MCLs** representados na **Figura 3 - 16** .

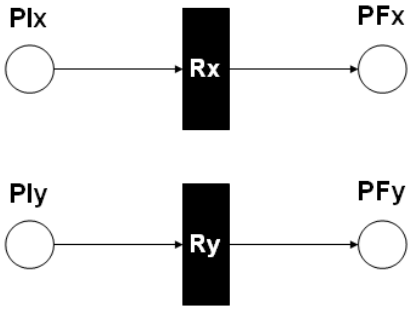


Figura 3 - 16 : **MCL** para o conjunto de relações **D**.

3.3.3.2. Mecanismo de Coordenação Global MCG

A coordenação das restrições temporais globais é realizada pelo mecanismo de coordenação global **MCG**. O Algoritmo 1 descreve essencialmente o processo de construção do **MCG**. O **MCG** é construído a partir dos mecanismos de coordenação das atividades, relações e estrelas. Estes mecanismos de coordenação são conectados ordenadamente (Algoritmo 1) obtendo-se ao final o **MCG** da expressão. Substituindo-se as caixas pretas deste **MCG** pelos **MCLs** correspondentes obtém-se o **MC** final da expressão.

Apresentam-se a seguir os diagramas para os mecanismos de coordenação dos elementos de uma expressão, isto é:

- mecanismo de coordenação para uma atividade,
- mecanismo de coordenação das relações em **D** (Definição 6),
- mecanismo de coordenação para o rótulo de aresta e
- mecanismo de coordenação para o rótulo de atividade.

Apresenta-se também o procedimento usado para efetuar a operação de conexão entre estes elementos.

a) Modelagem dos elementos de uma expressão

As restrições indiretas de uma expressão são modeladas usando redes de Petri coloridas (Colored Petri Nets – **CPNets**) ([Murata 89], [Jensen 97]). Dentre as justificativas para escolha deste modelo matemático evidenciam-se os benefícios das redes de Petri clássicas acrescidas de facilidades como a diferenciação entre fichas.

A diferenciação entre fichas é realizada criando-se tipos diferentes de fichas denominadas fichas coloridas.

A capacidade de diferenciar fichas é explorada essencialmente na seleção de relações e na seleção de atividades com a finalidade de obter-se uma rede mais compacta.

O modelo obtido pode ser traduzido, se necessário, para uma Rede de Petri clássica, pois o número de cores que uma ficha pode assumir é finito.

Modelagem de uma atividade

Da Definição 1 e Definição 6 tem-se que uma atividade $\mathbf{a_i}$:

i. tem um início e um fim;
ii. está relacionada a $\mathbf{a_j}$ por uma relação $\mathbf{r \in D}$;
iii. tanto o início quanto o fim de $\mathbf{a_i}$ podem estar sujeitos a restrições temporais.

Traduzindo estes itens para uma Rede de Petri Colorida chega-se ao diagrama **CPNet 1** ilustrado na Figura 3-17.

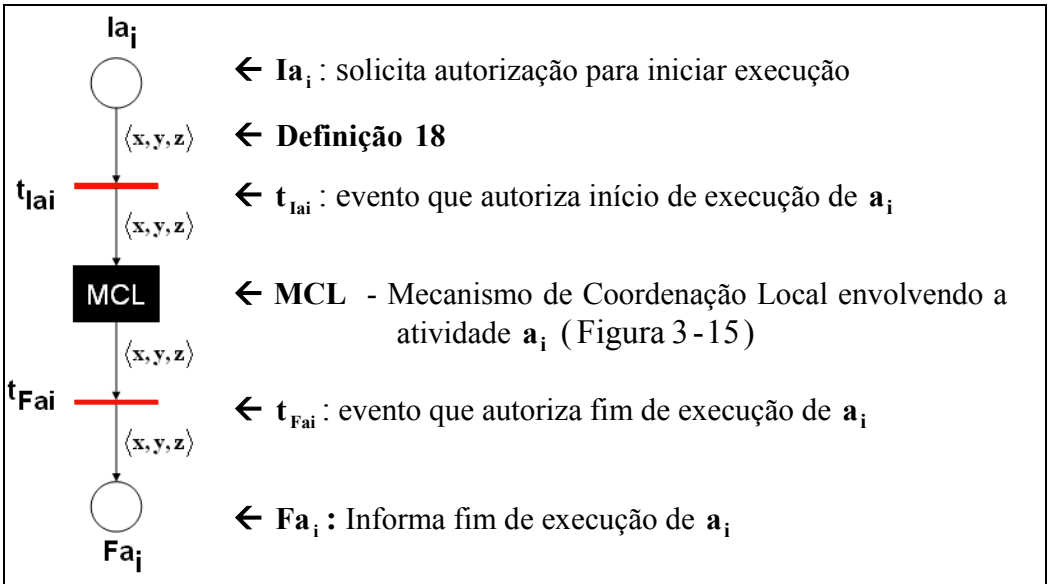


Figura 3-17: **CPNet 1** diagrama de uma atividade $\mathbf{a_i}$ no nível da Coordenação.

Para uma atividade ser executada ela deve receber uma solicitação de execução (do usuário, do sistema ou pelo evento início ou fim de uma atividade). Esta solicitação corresponde a uma ficha no lugar $\mathbf{Ia_i}$.

O disparo da transição $\mathbf{t_{Iai}}$ corresponde ao evento de autorização para o início da execução da atividade $\mathbf{a_i}$ e significa que todas as **CGs** impostas à $\mathbf{a_i}$ foram atendidas.

O Mecanismo de Coordenação Local **MCL** relativo à $\mathbf{a_i}$ (ver Seção 3.3.3.1) recebe o evento de autorização e:

i. ordena o início de $\mathbf{a_i}$;
ii. aguarda o fim da execução de $\mathbf{a_i}$.

O disparo da transição $\mathbf{t_{Fai}}$ gera o evento fim da execução de $\mathbf{a_i}$. Uma ficha no lugar $\mathbf{Fa_i}$ significa que $\mathbf{a_i}$ já foi executada.

Os arcos de entrada e de saída de uma transição são rotulados. Estes rótulos determinam a quantidade e o tipo de fichas (cor) que serão removidas de um lugar ou adicionadas a outro. O valor de cor que pode ser atribuído a uma ficha é um terno ordenado $\langle \mathbf{x,y,z} \rangle$ na forma estabelecida na Definição 18.

Definição 18: espaço de cores

> Uma cor é um terno ordenado $\langle \mathbf{x,y,z} \rangle$ tal que $(\mathbf{y,z}) \in \mathbf{R}$ e $\mathbf{x} \in \mathbf{F}(\mathbf{y,z})$, onde **R** é o conjunto de arestas de uma expressão e **F** a função rotuladora de arestas.

Ou seja, a variável **x** da cor $\langle \mathbf{x,y,z} \rangle$ especifica a relação entre o par ordenado de atividades **y**, **z**.

Por simplicidade de notação convenciona-se que a ausência de rótulo indica a remoção de uma ficha do lugar de entrada ou a adição de uma ficha ao lugar de saída, cujo tipo é o mesmo da ficha removida. Em caso de ambigüidade os rótulos são especificados explicitamente. Por exemplo, qualquer que seja a cor da ficha no lugar $\mathbf{Ia_i}$ de **CPNet 1** a transição $\mathbf{t_{Iai}}$ fica habilitada, esta situação pode ser descrita com a ausência de rótulo no arco $(\mathbf{Ia_i, t_{Iai}})$.

Modelagem de uma relação

Uma relação $\mathbf{r} \in \mathbf{D}$ entre duas atividades $\mathbf{a_j}$ e $\mathbf{a_k}$, $\mathbf{r}(\mathbf{a_j, a_k})$, é modelada a partir do diagrama **CPNet 1**, conforme mostra o diagrama **CPNet 2** ilustrado na Figura 3-18.

O diagrama **CPNet 2** combina os diagramas de $\mathbf{a_j}$ e $\mathbf{a_k}$ obtendo-se um modelo para **r**. As transições e lugares de **CPNet 2** continuam tendo o mesmo significado dos elementos de **CPNet 1**. A principal diferença é que agora o **MCL** da relação $\mathbf{r}(\mathbf{a_j, a_k})$ recebe os eventos de autorização de início tanto de $\mathbf{a_j}$ como de $\mathbf{a_k}$.

Para que o modelo **CPNet 2** apresentado na Figura 3-18 caracterize uma relação **r** é necessário adicionar-lhe as restrições diretas derivadas de **r**.

Analiticamente, uma restrição direta sempre envolve duas atividades distintas $\mathbf{a_j}$ e $\mathbf{a_k}$; é uma equação ou uma inequação formada por duas variáveis e um operador. As variáveis assumem valores no espaço temporal e representam o instante de início ou fim de execução das atividades. Os operadores usados na descrição de uma restrição são o operador de igualdade, o operador de desigualdade menor do que e o operador de desigualdade maior do que. Estes operadores são denotados pelos símbolos usuais $=$, $<$ e $>$.

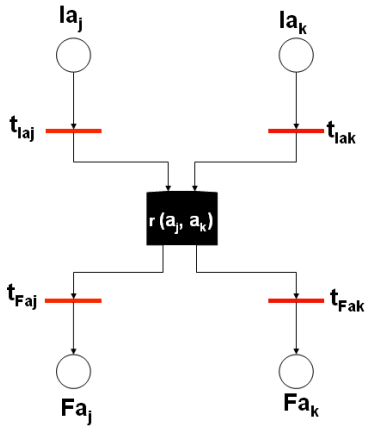


Figura 3 - 18: **CPnet2** diagrama da relação $\mathbf{r(a_j, a_k)}$

Uma restrição direta é adicionada à **CPNet 2** aplicando-se as regras **u1, u2, u3, u4** e **u5** apresentadas a seguir.

u1) O identificador de uma variável é formado pela letra minúscula **a** seguido do número **k**, $\mathbf{k \in \{1,2,\cdots,n\}}$ que identifica a atividade $\mathbf{a_k}$ acrescido de um sub-índice **i** ou **f**, conforme se queira representar o instante de tempo de início de execução $\mathbf{ak_i}$ ou de fim de execução $\mathbf{ak_f}$ da atividade $\mathbf{a_k}$;

u2) A variável $\mathbf{ak_i}$ deve ser associada a transição $\mathbf{t_{Iak}}$;

u3) A variável $\mathbf{ak_f}$ deve ser associada a transição $\mathbf{t_{Fak}}$;

u4) A inequação $\mathbf{x < y}$ ou $\mathbf{x > y}$, onde **x** e **y** representam os instantes de início ou fim de execução das atividades, é traduzida à **CPNet 2** adicionando um arco da transição que corresponde à variável de menor valor para o lugar $\mathbf{P_z}$, $\mathbf{z \in N}$, e um arco de $\mathbf{P_z}$ para transição que representa o maior valor;

u5) Adicionar uma equação à **CPNet 2** consiste em efetuar uma operação de fusão (ver Definição 20) das transições associadas às variáveis da equação.

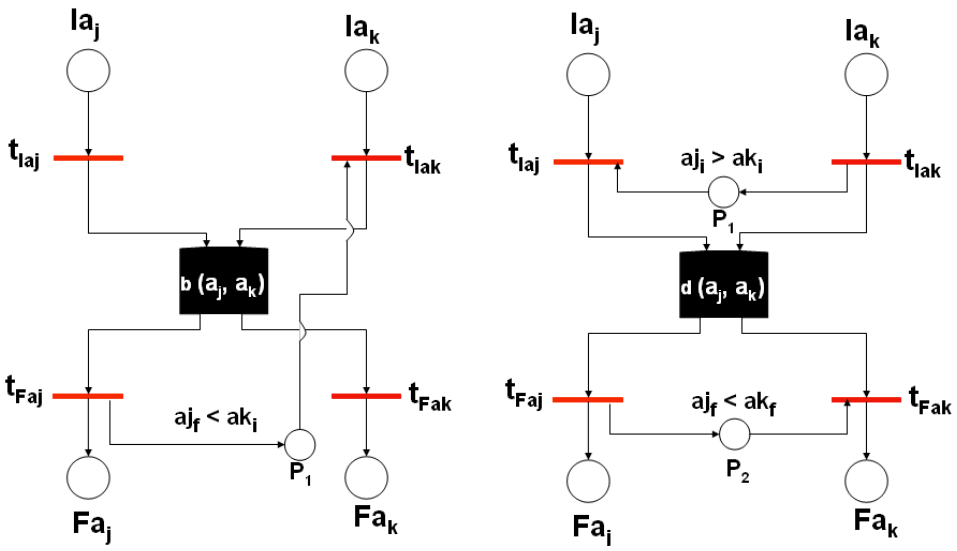
A seguir aplicam-se as regras **u1, u2, u3, u4** e **u5** ao diagrama **CPNet 2** para cada relação **r** de **D**. Ao fim de cada procedimento tem-se o mecanismo de coordenação **MC** de **r**.

Relação before: $\mathbf{b}(\mathbf{a}_j, \mathbf{a}_k)$

Da definição da relação "before" (Definição 06) deriva-se a restrição direta: $(\mathbf{aj}_f < \mathbf{ak}_i)$ (regra **u1**). A variável $\mathbf{aj}_f$ é associada a transição $\mathbf{t}_{\mathbf{Faj}}$ de $\mathbf{a}_j$ (regra **u3**) e $\mathbf{ak}_i$ é associada a transição (regra **u2**). Por fim a inequação $\mathbf{aj}_f < \mathbf{ak}_i$ é modelada pelos arcos $(\mathbf{t}_{\mathbf{Faj}}, \mathbf{P}_1)$ e $(\mathbf{P}_1, \mathbf{t}_{\mathbf{Iak}})$ conforme estabelece a regra **u4**. A Figura 3-19 (a) ilustra o diagrama da relação "before" entre $\mathbf{a}_j$ e $\mathbf{a}_k$.

Relação during: $\mathbf{d}(\mathbf{a}_j, \mathbf{a}_k)$

Da definição da relação "during" derivam-se as restrições $(\mathbf{aj}_i > \mathbf{ak}_i)$ e $(\mathbf{aj}_f < \mathbf{ak}_f)$ (regra **u1**) que são modeladas aplicando-se as regras **u2**, **u3**, e **u4**. A Figura 3-19 (b) ilustra o diagrama gerado para a relação $\mathbf{d}(\mathbf{a}_j, \mathbf{a}_k)$.



(a) MC da relação **before:** $\mathbf{b}(\mathbf{a}_j, \mathbf{a}_k)$. **(b) MC** da relação **during**: $\mathbf{d}(\mathbf{a}_j, \mathbf{a}_k)$

Figura 3-19: **MC** das relações **before** e **during**.

Relação overlaps: $\mathbf{o(a_j, a_k)}$

Da relação "overlaps" derivam-se as restrições $(\mathbf{aj_i < ak_i})$, $(\mathbf{ak_i < aj_f})$ e $(\mathbf{aj_f < ak_f})$ (regra **u1**). Aplicando-se as regras **u2**, **u3**, e **u4** chega-se ao diagrama apresentado na Figura 3 - 20 que corresponde ao mecanismo de coordenação gerado para relação $\mathbf{o(a_j, a_k)}$.

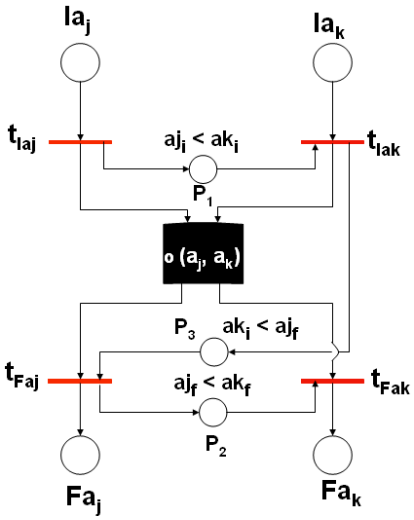


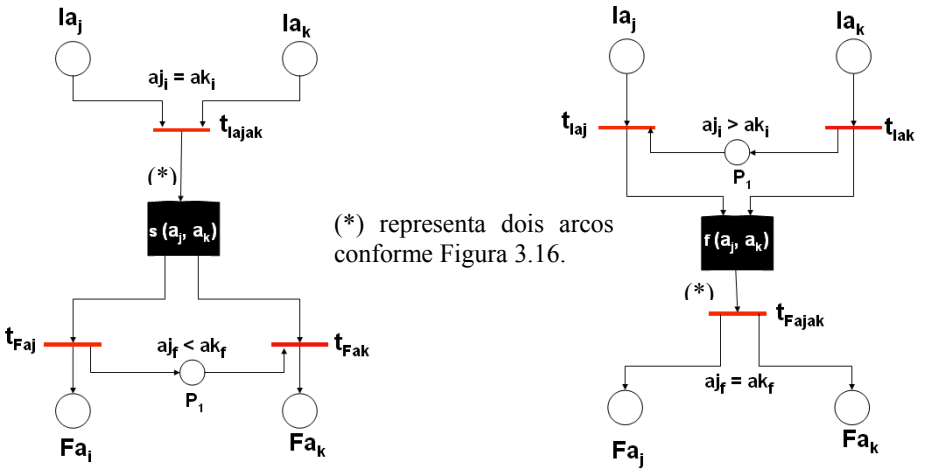
Figura 3 - 20 : **MC** da relação **overlaps**: $\mathbf{o(a_j, a_k)}$

Relação starts: $\mathbf{s(a_j, a_k)}$

Da relação "starts" derivam-se as restrições $(\mathbf{aj_i = ak_i})$ e $(\mathbf{aj_f < ak_f})$ (regra **u1**) que são modeladas aplicando-se as regras **u2**, **u3**, **u4** e **u5**. A condição de simultaneidade colocada pela restrição $(\mathbf{aj_i = ak_i})$ e tratada por meio da regra **u5**, determina a união das transições associadas as variáveis $\mathbf{aj_i}$ e $\mathbf{ak_i}$. Dessa forma os eventos de autorização para o início de $\mathbf{a_j}$ e $\mathbf{a_k}$ acontecem simultaneamente. A Figura 3 - 21 (a) apresenta o diagrama resultante para relação $\mathbf{s(a_j, a_k)}$.

Relação finishes: $\mathbf{f(a_j, a_k)}$

Da relação "finishes" derivam-se as restrições $(\mathbf{ak_i < aj_i})$ e $(\mathbf{aj_f = ak_f})$ (regra **u1**) e aplicando-se as regras **u2**, **u3**, **u4** e **u5** obtém-se o mecanismo de coordenação da relação $\mathbf{f(a_j, a_k)}$ apresentado na Figura 3 - 21 (b). A simultaneidade $(\mathbf{aj_f = ak_f})$ é tratada pela transição $\mathbf{t_{Fajak}}$.



(a) MC da relação **starts**: $\mathbf{s(a_j, a_k)}$. **(b) MC** da relação **finishes**: $\mathbf{f(a_j, a_k)}$.

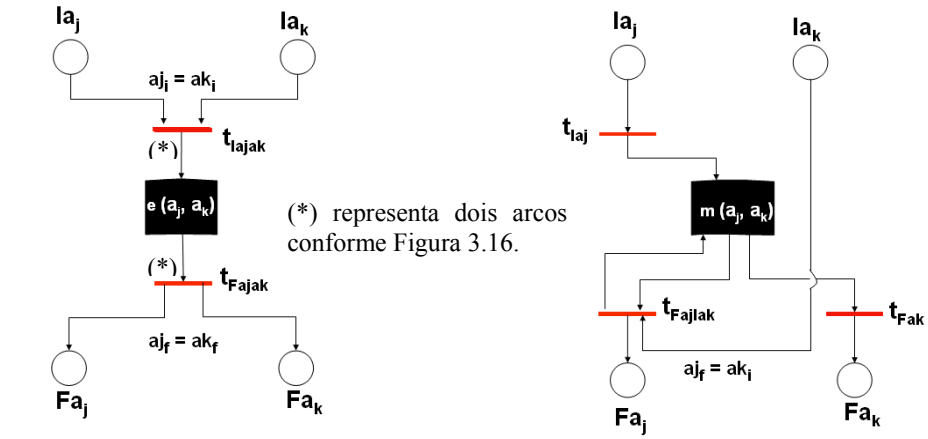
Figura 3 - 21 : **MC** das relações **starts** e **finishes**.

Relação equal: $\mathbf{e(a_j, a_k)}$

Da relação "equals" derivam-se as restrições $\mathbf{(ak_i = aj_i)}$ e $\mathbf{(aj_f = ak_f)}$ (regra **u1**) e aplicando-se as regras **u2**, **u3**, e **u5** obtém-se o **MC** da relação $\mathbf{e(a_j, a_k)}$ apresentado na Figura 3 - 22 (a). A simultaneidade $\mathbf{(ak_i = aj_i)}$ é tratada pela transição $\mathbf{t_{Iajak}}$ e a $\mathbf{(aj_f = ak_f)}$ pela transição $\mathbf{t_{Fajak}}$.

Relação meets: $\mathbf{m(a_j, a_k)}$

Da relação "meets" deriva-se a restrição $\mathbf{(aj_f = ak_i)}$(regra **u1**). Aplicando-se as regras **u2**, **u3**, e **u5** obtém-se o diagrama da Figura 3 - 22 (b). A simultaneidade $\mathbf{(aj_f = ak_i)}$ é tratada pela transição $\mathbf{t_{FajIak}}$.



(a) MC da relação **equal**: $\mathbf{e(a_j, a_k)}$. **(b) MC** da relação **meets**: $\mathbf{m(a_j, a_k)}$.

Figura 3 - 22 : **MC** das relações **equal** e **meets**.

Modelagem das relações especificadas em um rótulo de aresta não unitário

O rótulo $\mathbf{Y}_{j,k}$ da aresta $(\mathbf{a}_j, \mathbf{a}_k)$ (Seção 3.2.2) lista as primitivas através das quais as atividades $\mathbf{a}_j$ e $\mathbf{a}_k$ podem relacionar-se. Se **Y** é um conjunto unitário então um dos diagramas apresentados anteriormente pode ser usado para modelar a relação. Caso contrário faz-se necessário um modelo que permita selecionar um dos elemento de **Y** como sendo a relação entre $\mathbf{a}_j$ e $\mathbf{a}_k$. O diagrama **CPnet 3** apresentado na Figura 3 - 23 ilustra o **MC** para este caso.

Todas as relações listadas em **Y** são modeladas independentemente e de acordo com os mecanismos apresentados para cada relação de **D**. A relação de dependência entre $\mathbf{a}_j$ e $\mathbf{a}_k$ é selecionada segundo a cor da ficha. Por exemplo, para que a relação $\mathbf{r}_2$ seja selecionada deve-se ter uma ficha cor $\langle \mathbf{r}_2, \mathbf{a}_j, \mathbf{a}_k \rangle$ em **Iaj** e outra ficha cor $\langle \mathbf{r}_2, \mathbf{a}_j, \mathbf{a}_k \rangle$ em **Iak**.

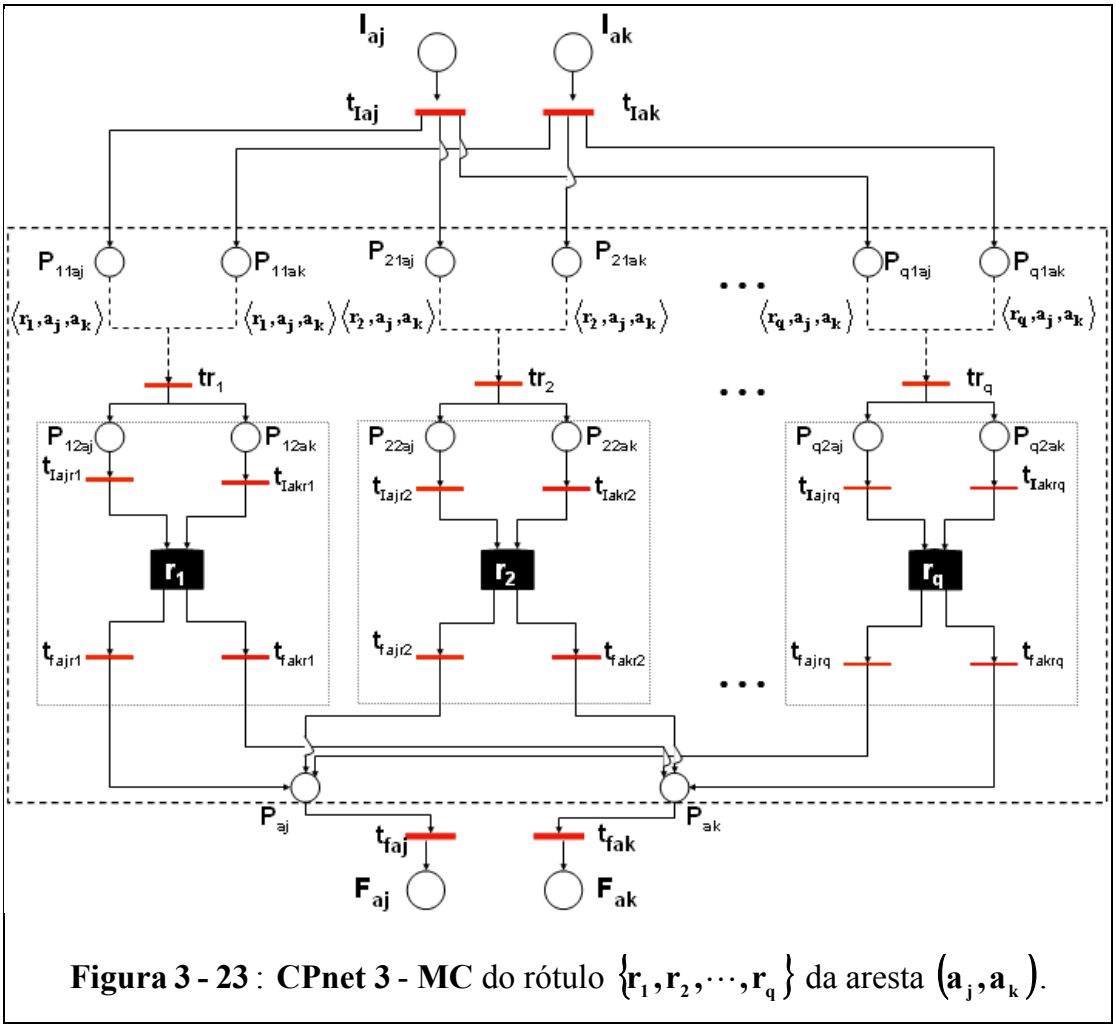


Figura 3 - 23 : **CPnet 3 - MC** do rótulo $\{\mathbf{r}_1, \mathbf{r}_2, \cdots, \mathbf{r}_q\}$ da aresta $(\mathbf{a}_j, \mathbf{a}_k)$.

Modelagem das relações de um rótulo de atividade

O modelo apresentado para o rótulo de uma atividade é análogo ao caso anterior, a diferença está na quantidade de atividades envolvidas, isto é, a atividade em questão mais todas as outras atividades do seu rótulo.

O modelo proposto para o rótulo de uma atividade é dado pelo diagrama **CPNet 4** ilustrado na Figura 3 - 24.

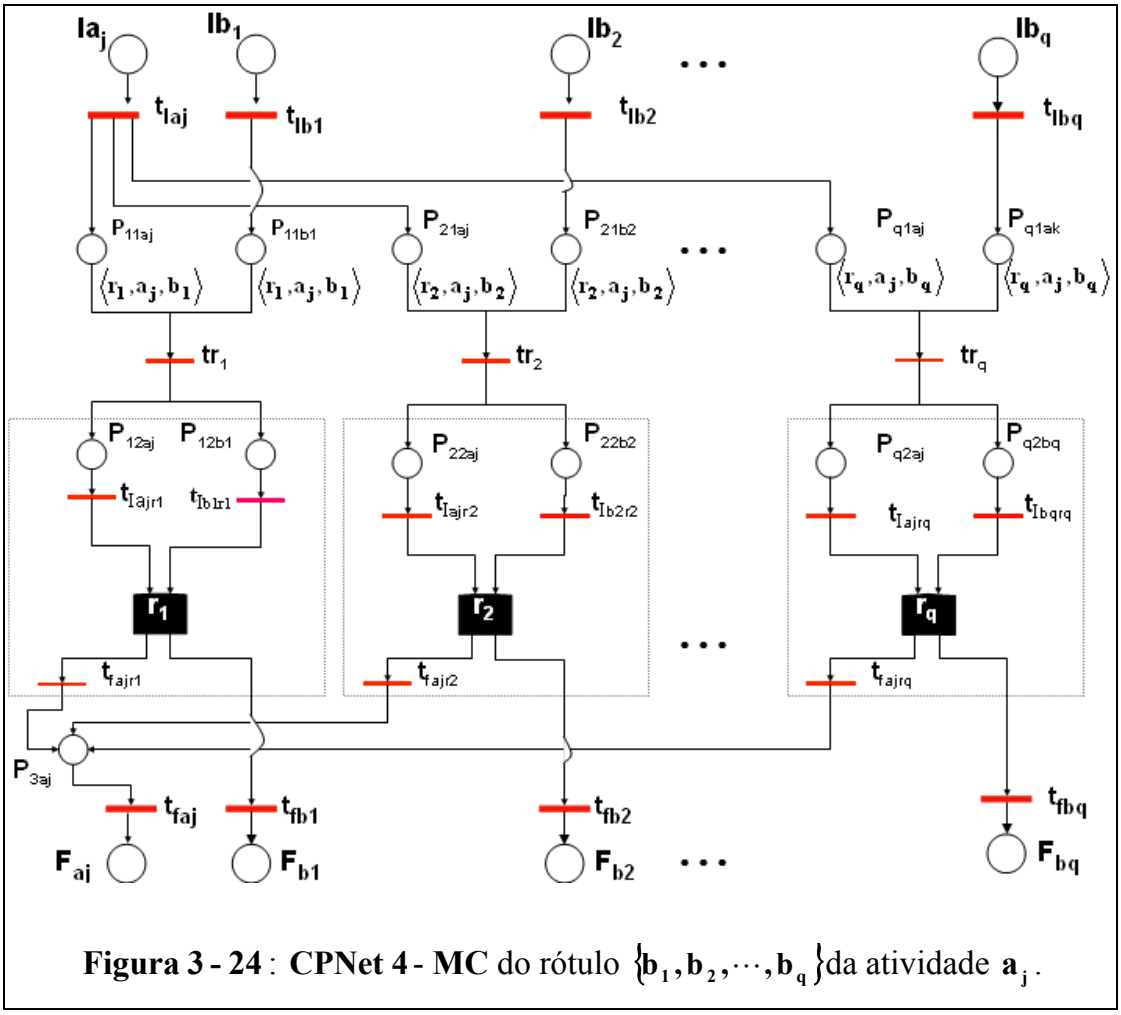


Figura 3 - 24: **CPNet 4 - MC** do rótulo $\{b_1, b_2, \cdots, b_q\}$ da atividade a_j.

O **CPNet 4** é composto pela coleção de **q** relações r_k, $k = 1,2,\cdots q$ entre a atividade a_j e as atividades b_k do seu rótulo, acrescido dos elementos responsáveis em promover a seleção de uma atividade b_k. O processo de seleção inicia com a requisição de execução de uma atividade, o que é caracterizado por uma ficha $\langle r_k, a_j, b_k \rangle$ em Ia_j ou Ib_k. O par de lugares Ia_j e Ib_k com a mesma cor de ficha determina a atividade b_k selecionada e a relação r_k entre a_j e b_k.

b) Operação de Conexão dos Mecanismos de Coordenação

Até este ponto foram apresentados os diagramas para se obter os mecanismos de coordenação para uma relação do conjunto **D** (Definição 6), o rótulo de uma aresta e o rótulo de uma atividade. Conectando-se apropriadamente esses elementos obtém-se o mecanismo de coordenação de uma estrela $\mathbf{a}_j$ (isto é, $\mathbf{MCa}_j$). Por exemplo, pode-se obter o **MC** da estrela $\mathbf{a}_3$ de $(\mathbf{E2, G2})$ (ver Figura $3-5$) conectando-se o **MC** da relação $\mathbf{b}(\mathbf{a}_3, \mathbf{a}_2)$ ao **MC** da relação $\mathbf{f}(\mathbf{a}_{11}, \mathbf{a}_3)$ e em seguida conectando-se o mecanismo resultante ao **MC** da relação $\mathbf{f}(\mathbf{a}_1, \mathbf{a}_3)$. Por fim, para obter-se o mecanismo de coordenação final de uma expressão precisa-se conectar os mecanismos de coordenação das estrelas.

A operação de conexão entre mecanismos de coordenação corresponde à conexão das listas de restrições temporais das expressões modeladas por estes.

A seguir começa-se definindo a operação de conexão entre dois **MC** e segue-se estendendo este procedimento até que todas as situações de conexão, listadas a seguir, tenham sido contempladas:

- Conexão de mecanismos derivados de relações (rótulos de arestas unitários);
- Conexão de mecanismos derivados de rótulo de arestas não unitários;
- Conexão de mecanismos derivados de rótulo de atividades;
- Conexão entre mecanismos de estrelas adjacentes

Definição 19: operação de conexão $\oplus$

> A operação $\oplus$, lê-se conexão, associa a cada par de mecanismos de coordenação $(\mathbf{MC}_1, \mathbf{MC}_2)$ o mecanismo de coordenação $\mathbf{MC}_1 \oplus \mathbf{MC}_2$ se existir uma atividade $\mathbf{a}_j$ comum a eles.

A operação de conexão baseia-se na seguinte condição de existência:

> dois mecanismos de coordenação $\mathbf{MC}_1$ e $\mathbf{MC}_2$, podem ser conectados se existir uma atividade $\mathbf{a}_j$ comum a eles, isto é, $\mathbf{a}_j$ está modelada tanto em um quanto em outro **MC**.

Conexão de mecanismos derivados de relações

Seja $\mathbf{a}_j$ a atividade comum a $\mathbf{MC}_1$ e $\mathbf{MC}_2$. Se $\mathbf{a}_j$ aparece nas duas redes então existem transições que autorizam o início e o fim de $\mathbf{a}_j$ tanto em uma como na outra. A **CPNet** resultante da operação de conexão é obtida efetuado-se uma *fusão* destas transições. Ou seja a transição que autoriza o início (fim) de $\mathbf{a}_j$ em $\mathbf{MC}_1$ e a transição que autoriza o início (fim) de $\mathbf{a}_j$ em $\mathbf{MC}_2$ são transformadas ("fundidas")

em uma transição na rede resultante. A Definição 20 formaliza o procedimento de fusão entre duas transições.

Definição 20: fusão de transições

A fusão de duas transições **t'** e **t"** é efetuada transferindo-se para uma delas todos os arcos de entrada e todos os de saída da outra.

Considerando o modelo proposto para atividade, tem-se que após a operação de fusão de transições existirão dois lugares de entrada (saída) solicitando o início (fim) de $\mathbf{a}_j$ na transição resultante **t'** (ou **t"**). Portanto um deles deve ser eliminado.

O Procedimento 1, a seguir, descreve como aplicar a operação de conexão $\oplus$ aos mecanismos $\mathbf{MC}_1$ e $\mathbf{MC}_2$.

Procedimento 1 (parcial): Conexão $\mathbf{MC}_1 \oplus \mathbf{MC}_2$

Sejam $\mathbf{MC}_1$ e $\mathbf{MC}_2$ os mecanismos que modelam as relações $\mathbf{r}(\mathbf{a}_i, \mathbf{a}_j)$ e $\mathbf{r'}(\mathbf{a}_j, \mathbf{a}_k)$ respectivamente:

a conexão $\mathbf{MC}_1 \oplus \mathbf{MC}_2$ é obtida efetuando-se uma fusão da transição que autoriza o início de $\mathbf{a}_j$ em $\mathbf{MC_1}$ com a transição que autoriza o início de $\mathbf{a}_j$ em $\mathbf{MC_2}$ e uma fusão da transição que autoriza o fim de $\mathbf{a}_j$ em $\mathbf{MC_1}$ com a transição que autoriza o fim de $\mathbf{a}_j$ em $\mathbf{MC_2}$.

A Figura 3 - 25 ilustra a aplicação da operação $\oplus$ entre os mecanismos $\mathbf{MC}_1$ e $\mathbf{MC}_2$ das relações $\mathbf{f(a1, a3)}$ e $\mathbf{b(a3, a2)}$, respectivamente. A operação está definida porque $\mathbf{a}_3$ é uma atividade comum às duas relações e portanto comuns aos seus mecanismos de coordenação. O mecanismo resultante ilustrado na Figura $3-25$ (c) é obtido efetuando-se:

- a fusão das transições que autorizam o início de $\mathbf{a}_3$: $\mathbf{t}_{\mathbf{Ia3}}$ de $\mathbf{MC}_1$ com $\mathbf{t}_{\mathbf{Ia3}}$ de $\mathbf{MC}_2$ obtendo-se $\mathbf{t}_{\mathbf{Ia3}}$ em $\mathbf{MC}_1 \oplus \mathbf{MC}_2$; e
- a fusão das transições que autorizam o fim de $\mathbf{a}_3$: $\mathbf{t}_{\mathbf{Fa3a1}}$ de $\mathbf{MC}_1$ com $\mathbf{t}_{\mathbf{Fa3}}$ de $\mathbf{MC}_2$ obtendo-se $\mathbf{t}_{\mathbf{Fa3a1}}$ em $\mathbf{MC}_1 \oplus \mathbf{MC}_2$.

Conexão de mecanismos derivados de rótulos de arestas não unitários.

O diagrama apresentado na Figura 3 - 23 corresponde ao mecanismo de coordenação do rótulo de uma aresta $(\mathbf{a}_j, \mathbf{a}_k)$. Mostra-se aqui como conectar este

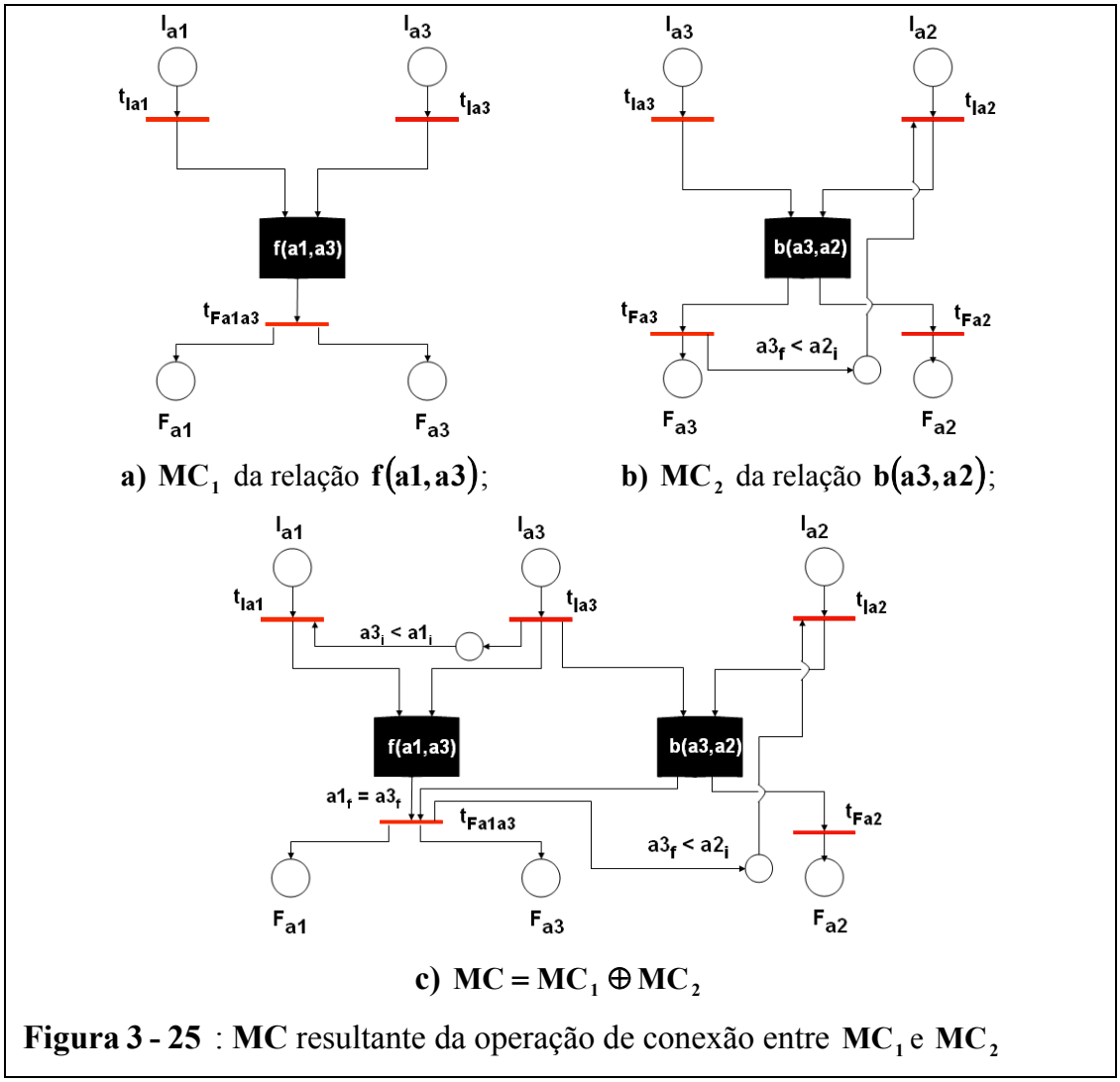
mecanismo a outros mecanismos de coordenação. A Figura 3 - 26 ilustra as três possibilidades de conexão.

Considerando-se, sem perda de generalidade, as arestas (a_i, a_j) e (a_j, a_k) da Figura 3 - 26 tem-se:

(a) o rótulo $\{r\}$ de (a_i, a_j) é unitário e o rótulo $\{r'\}$ de (a_j, a_k) é unitário;

(b) o rótulo $\{r\}$ de (a_i, a_j) é unitário e o rótulo $\{r'_1, r'_2, \cdots, r'_q\}$ de (a_j, a_k), para $q \geq 2$;

(c) o rótulo $\{r_1, r_2, \cdots, r_p\}$ de (a_i, a_j) para $p \geq 2$ e o rótulo $\{r'_1, r'_2, \cdots, r'_q\}$ de (a_j, a_k) para $q \geq 2$.



a) $\mathbf{MC_1}$ da relação $\mathbf{f(a1, a3)}$; **b)** $\mathbf{MC_2}$ da relação $\mathbf{b(a3, a2)}$;

c) $\mathbf{MC = MC_1 \oplus MC_2}$

Figura 3 - 25 : **MC** resultante da operação de conexão entre $\mathbf{MC_1}$ e $\mathbf{MC_2}$

A situação de conexão ilustrada na Figura 3 - 26 (a) é o caso relação / relação já tratado pelo Procedimento 1. Restam os casos ilustrados na Figura 3 - 26 (b) e

Figura 3 - 26 (c) que correspondem à conexão de mecanismos de coordenação derivados de relação / rótulo de aresta e rótulo de aresta / rótulo de aresta, respectivamente.

Seguindo a lógica do Procedimento 1, deve-se efetuar uma fusão das transições que autorizam o início da atividade comum $\mathbf{a_j}$ aos dois mecanismos de coordenação e uma fusão das transições que autorizam o fim da execução desta atividade. No entanto, do diagrama da Figura 3 - 23 sabe-se que existem duas transições, $\mathbf{t_{Iaj}}$ e $\mathbf{t_{Iajri}}$, $\mathbf{i \in \{1,2,\cdots,q\}}$, associadas ao evento autorizar início da execução de $\mathbf{a_j}$. Sendo que, a transição $\mathbf{t_{Iajri}}$ só é conhecida após selecionar-se uma das relações do rótulo da aresta $\mathbf{(a_j, a_k)}$, isto é, após o disparo de alguma das transições $\mathbf{t_{ri}}$, $\mathbf{i \in \{1,2,\cdots,q\}}$. Estes fatos mostram a necessidade de estender o Procedimento 1 de modo a atender aos casos de conexão já referidos. Com este objetivo estabelecem-se a seguir dois novos resultados.

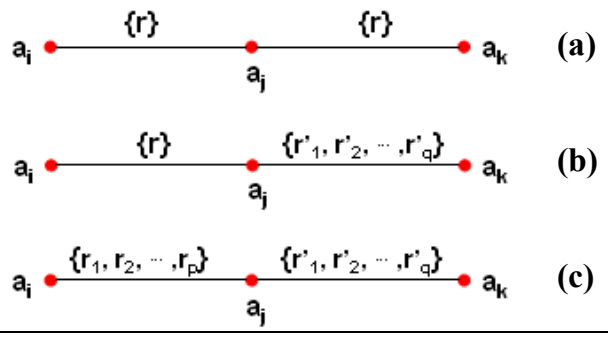


Figura 3 - 26: **(a)** 1x1, **(b)** 1xq e **(c)** pxq.

O primeiro resultado necessário para realizar a conexão entre mecanismos de coordenação derivados do caso relação / rótulo de aresta é a *fissão de transições* que permite cindir uma transição em duas outras.

Definição 21: fissão de transição

> A fissão da transição **t** nas transições $\mathbf{t_1}$ e $\mathbf{t_2}$ consiste em atribuir todos os arcos de entrada de **t** para $\mathbf{t_1}$ e todos os arcos de saída de **t** para $\mathbf{t_2}$.

O segundo resultado viabiliza os meios para conectar o diagrama da Figura 3 - 23 a outros mecanismos de coordenação. Para isto define-se o elemento dispositivo de conexão com a função de propagar para o mecanismo a ser conectado as restrições temporais derivadas da relação selecionada $\mathbf{r_i}$, $\mathbf{i \in \{1,2,\cdots,q\}}$ do rótulo da aresta $\mathbf{(a_j, a_k)}$.

Para adicionar este dispositivo de conexão ao diagrama da Figura 3 - 23 deve-se:

1. alterar o nome da transição $\mathbf{t_{Iaj}}$ para $\mathbf{t_{1Iaj}}$, denominada transição global *de* $\mathbf{a_j}$, e adiciona-se a transição $\mathbf{t_{2Iaj}}$, denominada transição local *de* $\mathbf{a_j}$.
2. alterar o nome da transição $\mathbf{t_{Iajri}}$, $\mathbf{i = 1,2,\cdots,q}$ para $\mathbf{t_{1ajri}}$ e adiciona-se a transição $\mathbf{t_{2ajri}}$,
3. adicionar os lugares $\mathbf{P_{1aj}}$ e $\mathbf{P_{2aj}}$.

A Figura 3 - 27 ilustra a adição do dispositivo de conexão ao diagrama da Figura 3 - 23, onde as linhas tracejadas representam os arcos rotulados explicitamente. Quanto à dinâmica desse dispositivo, o disparo da transição $\mathbf{t_{1Iaj}}$ indica que as restrições temporais impostas a $\mathbf{a_j}$ foram satisfeitas, a menos das restrições diretas derivadas da relação $\mathbf{r_i}$, $\mathbf{i \in \{1,2,\cdots,q\}}$ entre $\mathbf{a_j}$ e $\mathbf{a_k}$. O disparo da transição $\mathbf{t_{ri}}$ indica que a seleção de $\mathbf{r_i}$ já foi efetuada e o disparo da transição $\mathbf{t_{1Iajri}}$ indica que as restrições diretas derivadas de $\mathbf{r_i}$ e impostas a $\mathbf{a_j}$ já foram atendidas, promovendo assim o envio de uma ficha $\mathbf{< r_i, a_j, a_k >}$ ao lugar $\mathbf{P_{1aj}}$. O disparo da transição $\mathbf{t_{2Iajri}}$ habilita o início da execução de $\mathbf{a_j}$ em todos os mecanismos que coordenam relações de dependência com $\mathbf{a_j}$. O **MCL**, representado por uma caixa preta, correspondente à $\mathbf{r_i}$ providencia então o início de $\mathbf{a_j}$.

Se a atividade $\mathbf{a_k}$ também relaciona-se com outras atividades além de $\mathbf{a_j}$ então deve-se adicionar o dispositivo de conexão também para $\mathbf{a_k}$, conforme ilustra a Figura 3 - 28.

Finalmente, reescrevendo o Procedimento 1 de forma a incluir a conexão de **MCs** derivados de rótulos de arestas tem-se:

Procedimento 1(parcial): Conexão $\mathbf{MC_1 \oplus MC_2}$

Sejam $\mathbf{MC_1}$ e $\mathbf{MC_2}$ os mecanismos que modelam respectivamente as expressões $\boldsymbol{\varepsilon}$ e $\boldsymbol{\varepsilon'}$.

(a) Se o rótulo $\mathbf{Y_{i,j} = \{r\}}$ da aresta $\mathbf{(a_i, a_j)}$ é modelado por $\mathbf{MC_1}$ e o rótulo $\mathbf{Y_{j,k} = \{r'\}}$ da aresta $\mathbf{(a_j, a_k)}$ é modelado por $\mathbf{MC_2}$ a conexão $\mathbf{MC_1 \oplus MC_2}$ é obtida efetuando-se uma fusão da transição que autoriza o início de $\mathbf{a_j}$ em $\mathbf{MC_1}$ com a transição que autoriza o início de $\mathbf{a_j}$ em $\mathbf{MC_2}$ e uma fusão da transição que autoriza o fim de $\mathbf{a_j}$ em $\mathbf{MC_1}$ com a transição que autoriza o fim de $\mathbf{a_j}$ em $\mathbf{MC_2}$;

(b) Se o rótulo $\mathbf{Y_{i,j} = \{r\}}$ da aresta $\mathbf{(a_i, a_j)}$ é modelado por $\mathbf{MC_1}$ e o rótulo $\mathbf{Y_{j,k} = \{r'_1, r'_2, \cdots, r'_q\}}$ da aresta $\mathbf{(a_j, a_k)}$ é modelado por $\mathbf{MC_2}$ a conexão $\mathbf{MC_1 \oplus MC_2}$ é obtida efetuando-se uma fissão e uma fusão. A fissão é efetuada para a transição **t** que autoriza o início de execução de $\mathbf{a_j}$ em $\mathbf{MC_1}$ em duas transições que autorizam o início de execução de $\mathbf{a_j}$ em $\mathbf{MC_2}$, obedecendo-se a seguinte ordem: os arcos de entrada de **t** são atribuídos a transição global de $\mathbf{a_j}$ e os arcos de saída de **t** são atribuídos a transição local de $\mathbf{a_j}$. A fusão é efetuada entre a transição que autoriza o fim de $\mathbf{a_j}$ em $\mathbf{MC_1}$ com a transição que autoriza o fim de $\mathbf{a_j}$ em $\mathbf{MC_2}$;

(c) Se o rótulo $\mathbf{Y_{i,j} = \{r_1, r_2, \cdots, r_p\}}$ da aresta $\mathbf{(a_i, a_j)}$ é modelado por $\mathbf{MC_1}$ e o rótulo $\mathbf{Y_{j,k} = \{r'_1, r'_2, \cdots, r'_q\}}$ da aresta $\mathbf{(a_j, a_k)}$ é modelado por $\mathbf{MC_2}$ a conexão $\mathbf{MC_1 \oplus MC_2}$ é obtida efetuando-se uma fusão da transição global de $\mathbf{a_j}$ em $\mathbf{MC_1}$ com a transição global de $\mathbf{a_j}$ em $\mathbf{MC_2}$, uma fusão da transição local de $\mathbf{a_j}$ em $\mathbf{MC_1}$ com a transição local de $\mathbf{a_j}$ em $\mathbf{MC_2}$ e uma fusão da transição que autoriza o fim de $\mathbf{a_j}$ em $\mathbf{MC_1}$ com a transição que autoriza o fim de $\mathbf{a_j}$ em $\mathbf{MC_2}$.

Apresenta-se a seguir um exemplo da operação de conexão para o caso ilustrado na Figura 3 - 26 (b) considerando-se a expressão **E8(A,R,F)**, sendo:

$$\mathbf{A = \{a_1, a_2, a_3\},\ R = \{(a_2, a_1), (a_3, a_2)\}} \text{ e}$$

F a função rotuladora de arestas dada por

$$\mathbf{Y_{2,1} = \{b\}} \text{ e } \mathbf{Y_{3,2} = \{b, f\}}.$$

O mecanismo de coordenação de **E8(A,R,F)** é obtido conectando-se o mecanismo de coordenação $\mathbf{MC_1}$ do rótulo $\mathbf{Y_{2,1} = \{b\}}$ com o mecanismo de coordenação $\mathbf{MC_2}$ do rótulo $\mathbf{Y_{3,2} = \{b, f\}}$.

O $\mathbf{MC_1}$ (Figura 3 - 29) é obtido conforme a representação gráfica da Figura 3 - 19 (a) e o $\mathbf{MC_2}$ (Figura 3 - 30) é obtido do diagrama da Figura 3 - 27. O $\mathbf{MC_2}$ precisa de dispositivo de conexão apenas para atividade $\mathbf{a_2}$, pois $\mathbf{a_1}$ é uma atividade de grau 1, $\mathbf{\partial(a_1) = 1}$.

Por fim, o **MC** de **E8(A,R,F)**, ilustrado na Figura 3 - 31, é obtido efetuando-se a operação $\mathbf{MC_1 \oplus MC_2}$ através do item **(b)** do **Procedimento 1**. Primeiro efetua-se uma fissão da transição $\mathbf{t_{Ia2}}$ que autoriza o início de execução de $\mathbf{a_2}$ em $\mathbf{MC_1}$ nas transições $\mathbf{t_{1Ia2}}$ e $\mathbf{t_{2Ia2}}$ que autorizam o início de execução de $\mathbf{a_2}$ em $\mathbf{MC_2}$, obedecendo-se a seguinte ordem: os arcos de entrada de $\mathbf{t_{Ia2}}$ são atribuídos a transição $\mathbf{t_{1Ia2}}$ e os arcos de saída de $\mathbf{t_{Ia2}}$ são atribuídos a transição $\mathbf{t_{2Ia2}}$. A fusão é

efetuada entre a transição $\mathbf{t_{fa2}}$ que autoriza o fim de $\mathbf{a_2}$ em $\mathbf{MC_1}$ com a transição $\mathbf{t_{fa2}}$ que autoriza o fim de $\mathbf{a_2}$ em $\mathbf{MC_2}$.

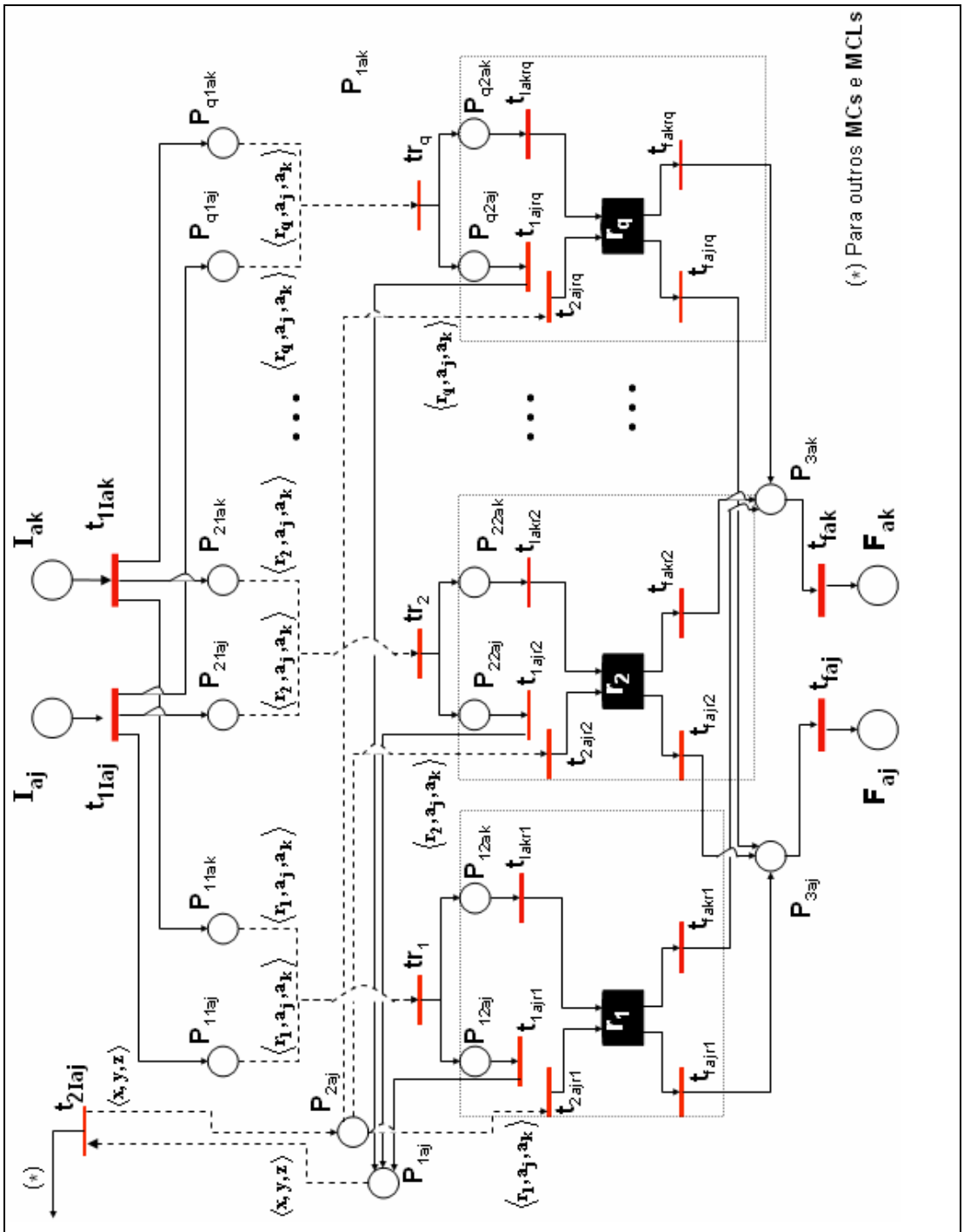


Figura 3 - 27 : **MC** do rótulo de $(\mathbf{a_j}, \mathbf{a_k})$ com dispositivo de conexão para $\mathbf{a_j}$.

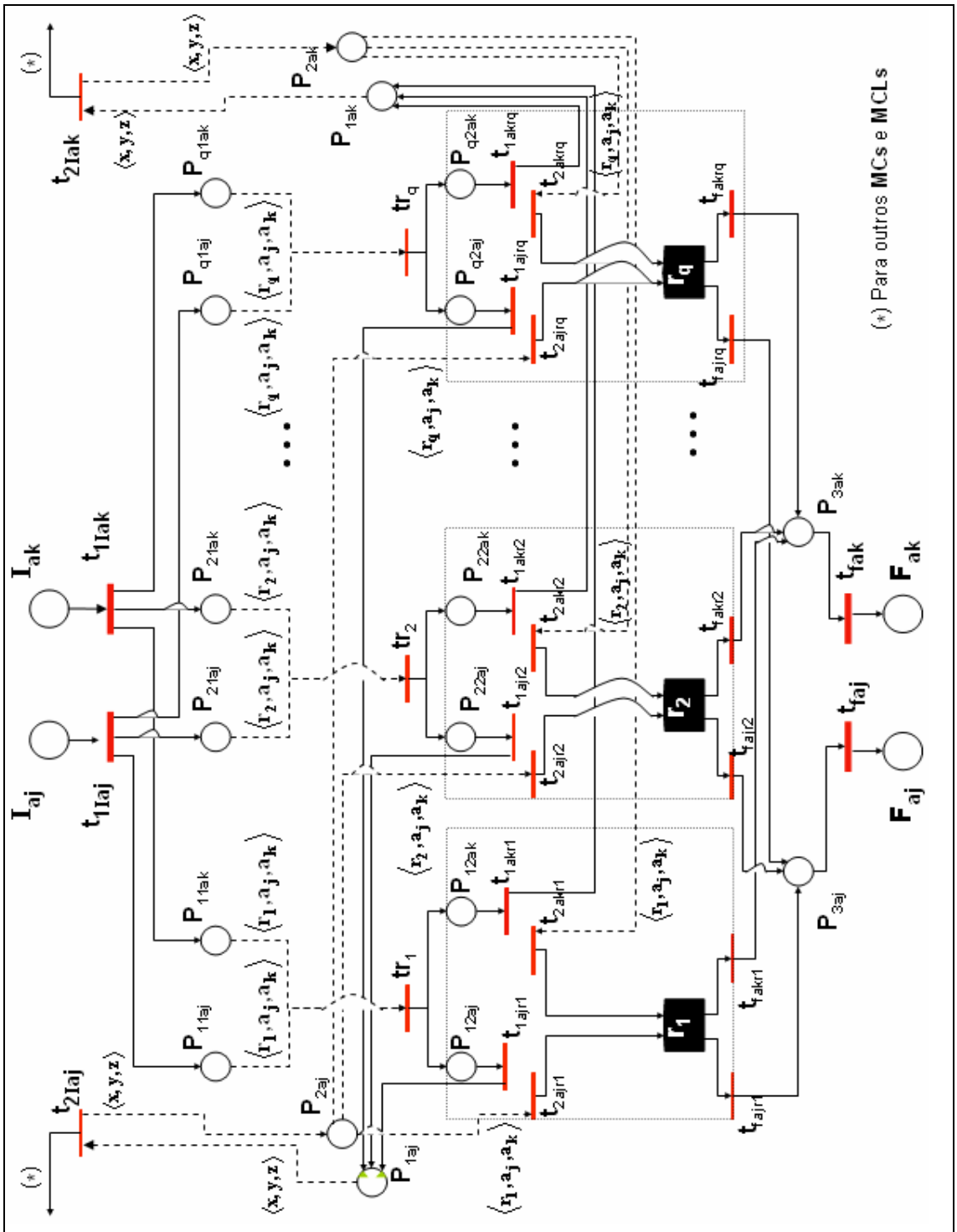


Figura 3 - 28: **MC** do rótulo de $(\mathbf{a}_j, \mathbf{a}_k)$ com dispositivo de conexão para $\mathbf{a}_j$ e $\mathbf{a}_k$.

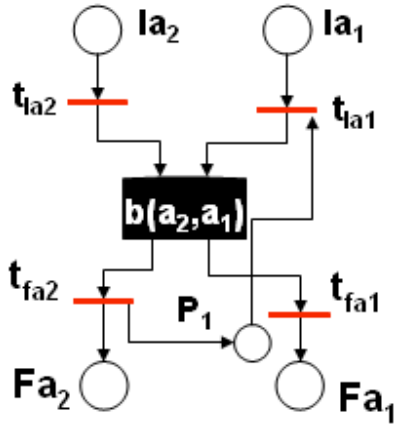


Figura 3 - 29 : MC_1 mecanismo do rótulo $Y_{2,1} = \{b\}$.

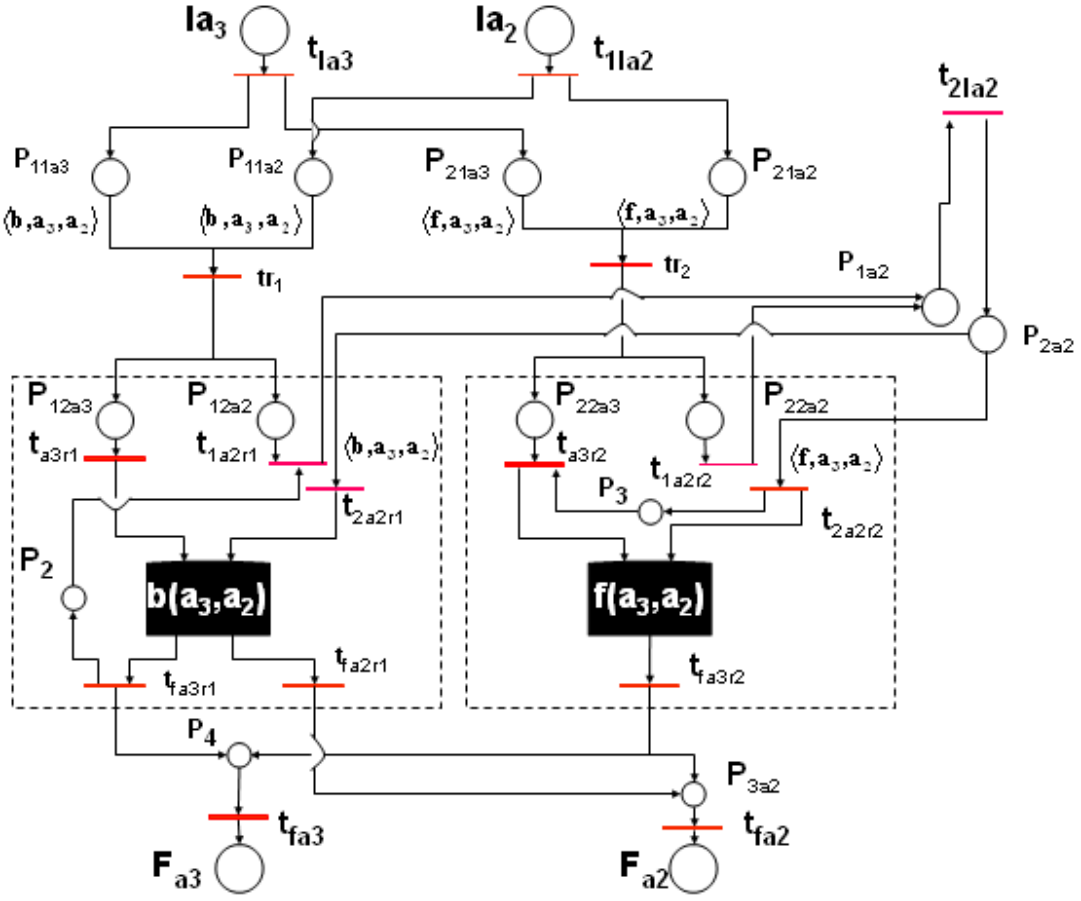


Figura 3 - 30 : MC_2 mecanismo do rótulo $Y_{3,2} = \{b,f\}$.

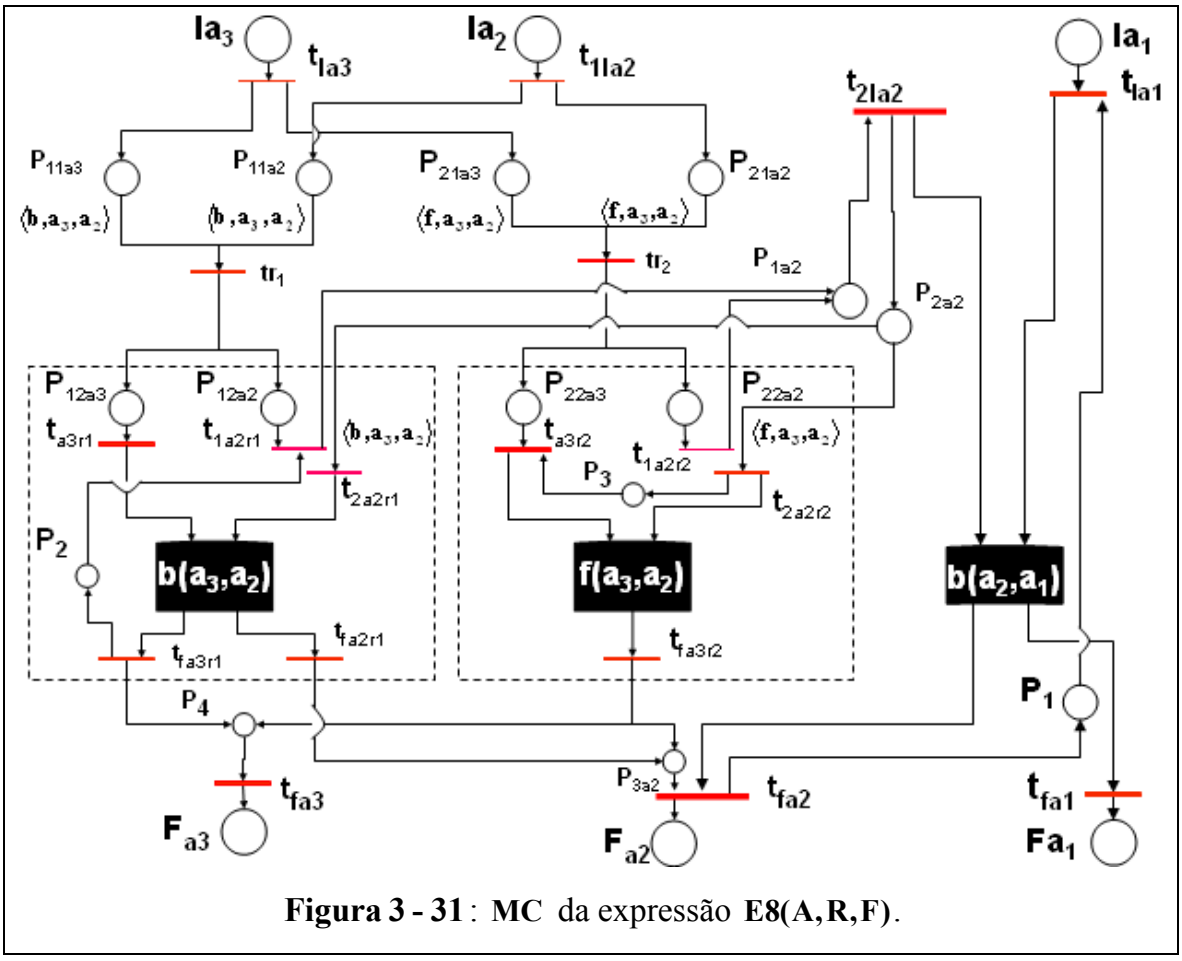


Figura 3 - 31: **MC** da expressão **E8(A,R,F)**.

Conexão de mecanismos derivados de rótulos de atividades

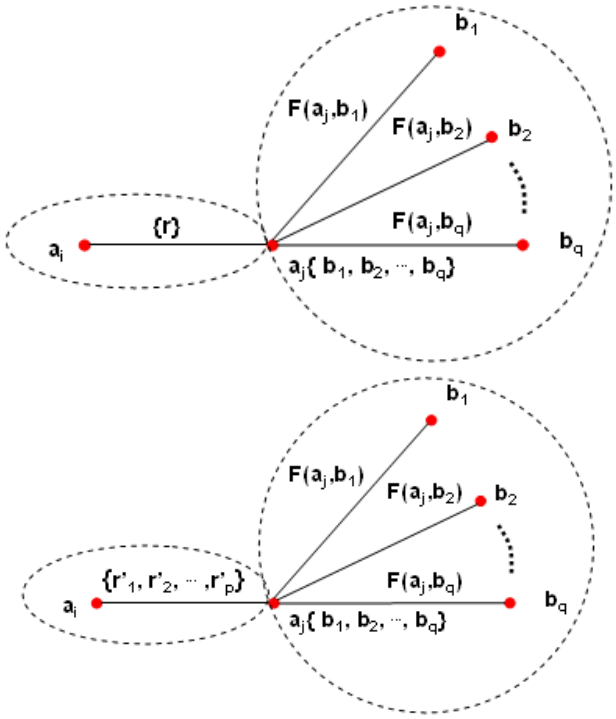
Mostra-se aqui como conectar o mecanismo de coordenação do rótulo de uma atividade $\mathbf{a_j}$ (Figura 3 - 24) a outro mecanismo de coordenação. Com este objetivo considere-se o mecanismo $\mathbf{MC_1}$ derivado da aresta $(\mathbf{a_j},\mathbf{a_i})$ e o mecanismo $\mathbf{MC_2}$ derivado do rótulo $\mathbf{X_j}=\{\mathbf{b_1},\mathbf{b_2},\cdots,\mathbf{b_q}\}$ da atividade $\mathbf{a_j}$. As possibilidades de conexão entre $\mathbf{MC_1}$ e $\mathbf{MC_2}$, ilustradas na **Figura 3 - 32** são a e b, respectivamente:

(a) a aresta $(\mathbf{a_j},\mathbf{a_i})$ tem rótulo unitário $\mathbf{Y_{j,i}}=\{\mathbf{r}\}$,

(b) a aresta $(\mathbf{a_j},\mathbf{a_i})$ tem rótulo com mais de uma relação $\mathbf{Y_{j,i}}=\{\mathbf{r_1},\mathbf{r_2},\cdots,\mathbf{r_p}\}$.

Como a atividade $\mathbf{a_j}$ está modelada tanto em $\mathbf{MC_1}$ como em $\mathbf{MC_2}$ a operação $\mathbf{MC_1}\oplus\mathbf{MC_2}$ pode ser efetuada. Do diagrama da Figura 3 - 24 observa-se, como feito para conexão de **MC** derivado de rótulo de aresta, os fatos de existirem duas transições, $\mathbf{t_{Iaj}}$ e $\mathbf{t_{Iajri}}$, $\mathbf{i}\in\{\mathbf{1,2,\cdots,q}\}$ associadas ao evento autorizar início da

execução de $\mathbf{a_j}$ e da transição $\mathbf{t_{Iajri}}$ ser conhecida só após a seleção de uma das atividades do rótulo de $\mathbf{a_j}$. Portanto, adota-se aqui a mesma solução adotada para conexão de um mecanismo derivado do rótulo de uma aresta, isto é, o uso de dispositivo de conexão.



(a) rótulo unitário

(b) rótulo com mais de uma relação

Figura 3 - 32 : Conexão de mecanismo derivado de rótulo de atividade.

A Figura 3 - 33 ilustra o mecanismo de coordenação para o rótulo de uma atividade, obtido do diagrama da Figura 3 - 24 adicionando-se o dispositivo de conexão à $\mathbf{a_j}$ da forma como segue:

1. altera-se o nome da transição $\mathbf{t_{Iaj}}$ para $\mathbf{t_{1Iaj}}$, denominada *transição global de* $\mathbf{a_j}$, e adiciona-se a transição $\mathbf{t_{2Iaj}}$, denominada *transição local de* $\mathbf{a_j}$.
2. altera-se o nome da transição $\mathbf{t_{Iajri}}$, $\mathbf{i = 1,2,\cdots,q}$ para $\mathbf{t_{1Iajri}}$ e adiciona-se a transição $\mathbf{t_{2Iajri}}$,
3. adicionam-se os lugares $\mathbf{P_{1aj}}$ e $\mathbf{P_{2aj}}$.

A dinâmica do dispositivo de conexão permanece a mesma anteriormente definida, ou seja, o disparo da transição $\mathbf{t_{1Iaj}}$ indica que as restrições temporais

impostas a $\mathbf{a}_j$ foram satisfeitas, a menos das restrições diretas derivadas da relação entre $\mathbf{a}_j$ e a atividade $\mathbf{b}_k$, $\mathbf{k} \in \{1,2,\cdots,q\}$.

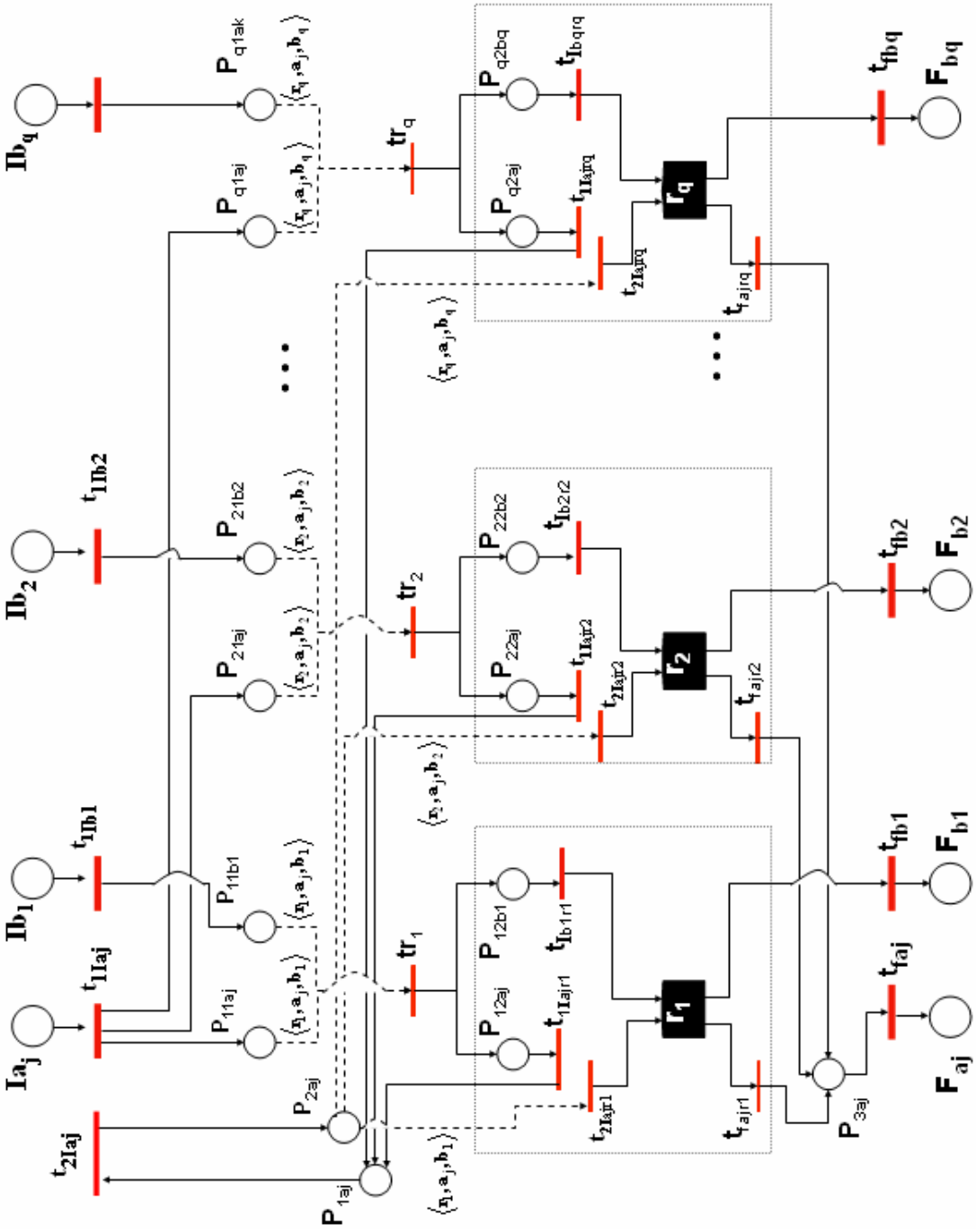


Figura 3 - 33 : **MC** do rótulo de $\mathbf{a}_j$ com dispositivo de conexão para $\mathbf{a}_j$

O disparo da transição $\mathbf{t_{ri}}$, $\mathbf{i \in \{1,2,\cdots,q\}}$ corresponde a seleção de uma das atividades $\mathbf{b_k}$ e o disparo da transição $\mathbf{t_{1Iajri}}$ promove o envio de uma ficha $\mathbf{< r_i, a_j, b_k >}$ ao lugar $\mathbf{P_{1aj}}$ indicando que as restrições diretas derivadas de $\mathbf{r_i}$ e impostas a $\mathbf{a_j}$ já foram atendidas. O disparo da transição $\mathbf{t_{2Iajri}}$ habilita o início da execução de $\mathbf{a_j}$ em todos os mecanismos locais que coordenam relações de dependência com $\mathbf{a_j}$. O **MCL** correspondente à $\mathbf{r_i}$ providencia então o início de $\mathbf{a_j}$.

Se as atividades $\mathbf{b_k}$ também relacionam-se com outras atividades ($\mathbf{\partial(b_k) > 1}$) além de $\mathbf{a_j}$ então deve-se adicionar os dispositivos de conexão também para estas atividades. A Figura 3 - 34 ilustra a forma mais complexa, onde todas as atividades $\mathbf{b_k, k = 1,2,\cdots,q}$, têm grau maior do que um.

Adicionado os dispositivos de conexão pode-se efetuar a operação de conexão para os casos representados na Figura 3 - 32. Para estes casos não é necessário estender o Procedimento 1 visto que os dispositivos de conexão não foram alterados.

A situação ilustrada na Figura 3 - 32 (a) representa a conexão entre o mecanismo $\mathbf{MC_1}$ que modela o rótulo unitário $\mathbf{Y_{j,i} = \{r\}}$ da aresta $\mathbf{(a_j, a_i)}$ e o mecanismo $\mathbf{MC_2}$ que modela o rótulo $\mathbf{X_j = \{b_1, b_2, \cdots, b_q\}}$ da atividade $\mathbf{a_j}$. Este caso é efetuado aplicando-se o item **(b)** do Procedimento 1**,** pois existe uma transição **t** que autoriza o início de execução de $\mathbf{a_j}$ em $\mathbf{MC_1}$ (Figura 3 - 18) e as duas transições que autorizam o início de execução de $\mathbf{a_j}$ em $\mathbf{MC_2}$ (Figura 3 - 33).

A situação ilustrada na Figura 3 - 32 (b) representa a conexão entre o mecanismo $\mathbf{MC_1}$ que modela o rótulo não unitário $\mathbf{Y_{j,i} = \{r_1, r_2, \cdots, r_p\}}$ da aresta $\mathbf{(a_j, a_i)}$ e o mecanismo $\mathbf{MC_2}$ que modela o rótulo $\mathbf{X_j = \{b_1, b_2, \cdots, b_q\}}$ da atividade $\mathbf{a_j}$. Este caso é efetuado aplicando-se o item **(c)** do Procedimento 1**,** pois existem duas transições que autorizam o início de execução de $\mathbf{a_j}$ em $\mathbf{MC_1}$ (Figura 3 - 27) e duas transições que autorizam o início de execução de $\mathbf{a_j}$ em $\mathbf{MC_2}$ (Figura 3 - 33).

O Procedimento 1 na sua versão final passa a ter o seguinte texto:

Procedimento 1: Conexão $\mathbf{MC_1 \oplus MC_2}$

Sejam $\mathbf{MC_1}$ e $\mathbf{MC_2}$ os mecanismos que modelam respectivamente as expressões $\boldsymbol{\varepsilon}$ e $\boldsymbol{\varepsilon'}$.

(a) Se o rótulo $\mathbf{Y_{i,j} = \{r\}}$ da aresta $\mathbf{(a_i, a_j)}$ está modelado em $\mathbf{MC_1}$ e o rótulo $\mathbf{Y_{j,k} = \{r'\}}$ da aresta $\mathbf{(a_j, a_k)}$ está modelado em $\mathbf{MC_2}$ a

conexão $\mathbf{MC_1} \oplus \mathbf{MC_2}$ é obtida efetuando-se uma fusão da transição que autoriza o início de $\mathbf{a_j}$ em $\mathbf{MC_1}$ com a transição que autoriza o início de $\mathbf{a_j}$ em $\mathbf{MC_2}$ e uma fusão da transição que autoriza o fim de $\mathbf{a_j}$ em $\mathbf{MC_1}$ com a transição que autoriza o fim de $\mathbf{a_j}$ em $\mathbf{MC_2}$;

(b) Se o rótulo $\mathbf{Y_{i,j}} = \{\mathbf{r}\}$ da aresta $(\mathbf{a_i}, \mathbf{a_j})$ está modelado em $\mathbf{MC_1}$ e o rótulo $\mathbf{Y_{j,k}} = \{\mathbf{r'_1}, \mathbf{r'_2}, \cdots, \mathbf{r'_q}\}$ da aresta $(\mathbf{a_j}, \mathbf{a_k})$ ou o rótulo $\mathbf{X_j} = \{\mathbf{b_1}, \mathbf{b_2}, \cdots, \mathbf{b_q}\}$ da atividade $\mathbf{a_j}$ estão modelados em $\mathbf{MC_2}$ a conexão $\mathbf{MC_1} \oplus \mathbf{MC_2}$ é obtida efetuando-se uma fissão e uma fusão. A fissão é efetuada na transição **t** que autoriza o início de execução de $\mathbf{a_j}$ em $\mathbf{MC_1}$ nas duas transições que autorizam o início de execução de $\mathbf{a_j}$ em $\mathbf{MC_2}$, obedecendo-se a seguinte ordem: os arcos de entrada de **t** são atribuídos a transição global de $\mathbf{a_j}$ e os arcos de saída de **t** são atribuídos a transição local de $\mathbf{a_j}$. A fusão é efetuada entre a transição que autoriza o fim de $\mathbf{a_j}$ em $\mathbf{MC_1}$ com a transição que autoriza o fim de $\mathbf{a_j}$ em $\mathbf{MC_2}$;

(c) Se o rótulo $\mathbf{Y_{i,j}} = \{\mathbf{r_1}, \mathbf{r_2}, \cdots, \mathbf{r_p}\}$ da aresta $(\mathbf{a_i}, \mathbf{a_j})$ está modelado em $\mathbf{MC_1}$ e o rótulo $\mathbf{Y_{j,k}} = \{\mathbf{r'_1}, \mathbf{r'_2}, \cdots, \mathbf{r'_q}\}$ da aresta $(\mathbf{a_j}, \mathbf{a_k})$ ou o rótulo $\mathbf{X_j} = \{\mathbf{b_1}, \mathbf{b_2}, \cdots, \mathbf{b_q}\}$ da atividade $\mathbf{a_j}$ estão modelados em $\mathbf{MC_2}$ a conexão $\mathbf{MC_1} \oplus \mathbf{MC_2}$ é obtida efetuando-se uma fusão da transição global de $\mathbf{a_j}$ em $\mathbf{MC_1}$ com a transição global de $\mathbf{a_j}$ em $\mathbf{MC_2}$, uma fusão da transição local de $\mathbf{a_j}$ em $\mathbf{MC_1}$ com a transição local de $\mathbf{a_j}$ em $\mathbf{MC_2}$ e uma fusão da transição que autoriza o fim de $\mathbf{a_j}$ em $\mathbf{MC_1}$ com a transição que autoriza o fim de $\mathbf{a_j}$ em $\mathbf{MC_2}$.

Um exemplo da operação de conexão envolvendo o mecanismo de coordenação derivado do rótulo de atividade é apresentado em detalhes na Seção 4.1.

Estabelecido o procedimento de conexão entre os diferentes tipos de mecanismos de coordenação pode-se agora gerar o mecanismo de coordenação de uma estrela $\mathbf{a_j}$ (Definição 15). Este mecanismo corresponde a modelagem das restrições temporais da estrela $\mathbf{a_j}$ (Definição 16) e é obtido conectando-se os mecanismos de coordenação dos elementos que compõem essa estrela, isto é, os

mecanismos das relações de $\mathbf{a_j}$ com outras atividades, os mecanismos derivados de arestas rotuladas e o mecanismo derivado do rótulo de $\mathbf{a_j}$. Escrevendo-se na forma de procedimento tem-se:

Procedimento 2: mecanismo de coordenação da estrela $\mathbf{a_j}$

1. gerar o mecanismo $\mathbf{MC_1}$ correspondente ao rótulo de $\mathbf{a_j}$;
2. gerar os mecanismos $\mathbf{MC_2, MC_3, \cdots, MC_p}$ para os rótulos das arestas definidas pelas atividades $\mathbf{a_j}$ e $\mathbf{a_i}$, se $\mathbf{a_i}$ satisfaz as seguintes propriedades:
   a. $\mathbf{a_i}$ não pertence ao rótulo de $\mathbf{a_j}$ ($\mathbf{a_i \notin G(a_j)}$) e
   b. $\mathbf{a_i}$ é uma atividade da curva de nível 0 (Definição 14), $\mathbf{a_i \in I_0}$;
3. gerar o mecanismo $\mathbf{MC_{p+1}}$ correspondente ao rótulo da aresta definida pelas atividades $\mathbf{a_j}$ e $\mathbf{a_i}$, se uma das propriedades a seguir for satisfeita:
   a. $\mathbf{a_j \notin G(a_i)}$, $\mathbf{a_i \notin G(a_j)}$, $\mathbf{a_j \in I_k}$ e $\mathbf{a_i \in I_{k+1}}$, $\mathbf{k > 0}$;
   b. $\mathbf{a_j \notin G(a_i)}$, $\mathbf{a_i \notin G(a_j)}$, $\mathbf{a_j}$ e $\mathbf{a_i}$ são as duas atividades que definem o centro (Seção 3.3.2) da expressão e o rótulo da aresta definida por $\mathbf{a_j}$ e $\mathbf{a_i}$ não foi modelado no **MC** da estrela $\mathbf{a_i}$;
   c. $\mathbf{a_j}$ é a única atividade do centro (Seção 3.3.2) da expressão e $\mathbf{MC_{p+1}}$ corresponde à sua modelagem (Figura 3-17).
4. gerar o mecanismo da estrela $\mathbf{a_j}$ efetuando-se as seguintes operações de conexão:

$$\mathbf{MCa_j = ((\cdots((MC_1 \oplus MC_2) \oplus MC_3) \oplus \cdots) \oplus MC_p) \oplus MC_{p+1}}$$

As propriedades do passo 2 (a. e b.) e passo 3 (a., b. e c.) têm por objetivo assegurar que o mecanismo de coordenação derivado da aresta entre duas estrelas adjacentes (Definição 17) seja gerado uma única vez.

A ordem em que as operações $\oplus$ são efetuadas pode ser alterada, pois ela satisfaz as propriedades comutativa e associativa:

i. $\mathbf{(MC_1 \oplus MC_2) = (MC_2 \oplus MC_1)}$;
ii. $\mathbf{(MC_1 \oplus MC_2) \oplus MC_3 = MC_1 \oplus (MC_2 \oplus MC_3)}$

Estas propriedades são verificadas efetuando-se a operação de conexão entre os diagramas (**CPNet 1**, **CPNet 2**, **CPNet 3** e **CPNet 4**) apresentados no decorrer da Seção 3.3.3.2. e constatando-se a equivalência entre eles. Exemplos de uso do Procedimento 2 são apresentados no Capítulo 4.

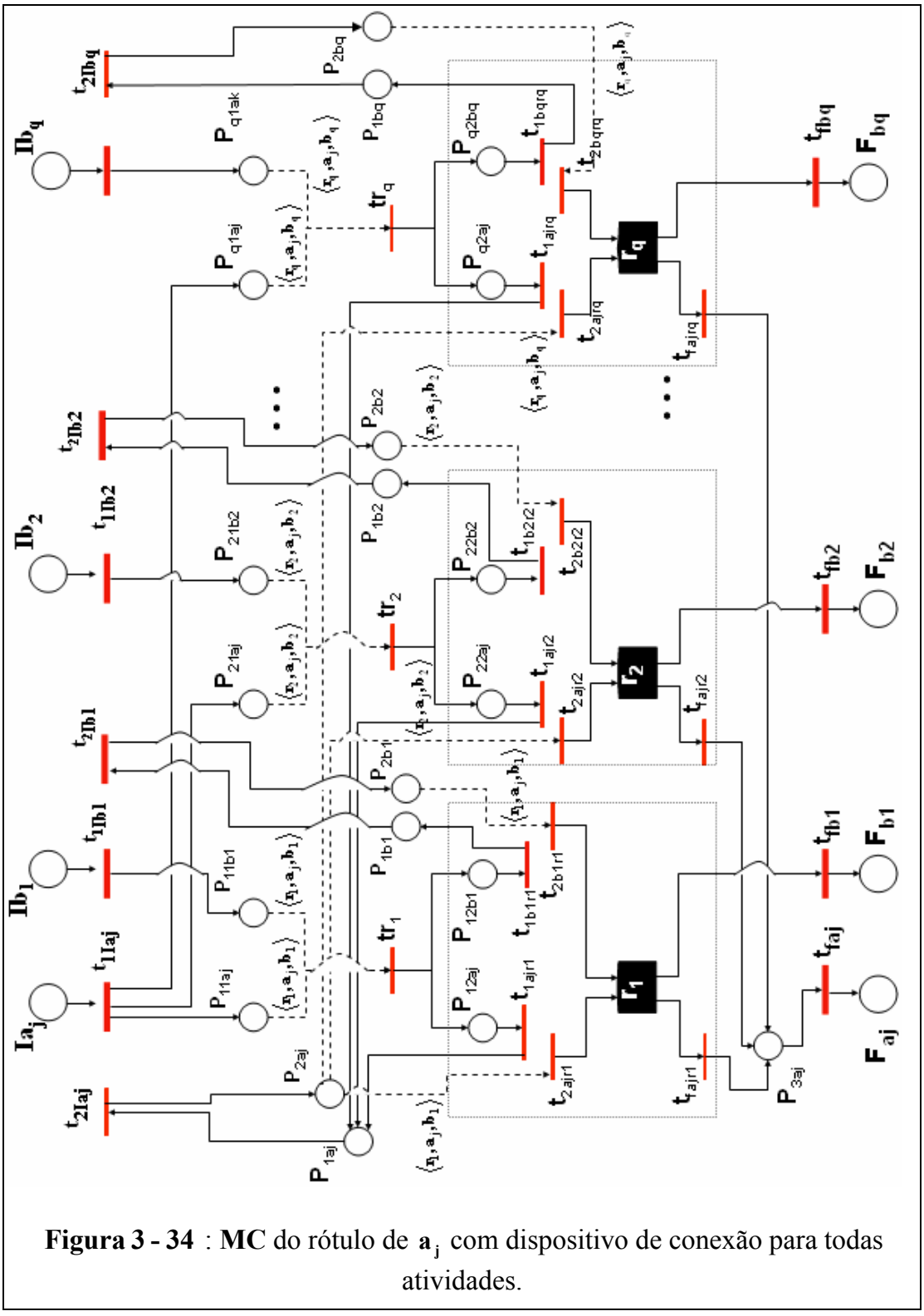


Figura 3 - 34 : **MC** do rótulo de $\mathbf{a}_j$ com dispositivo de conexão para todas atividades.

c) Conexão entre mecanismos de estrelas adjacentes

Para obter-se o mecanismo de coordenação final de uma expressão resta conectar os mecanismos de coordenação das estrelas na ordem estabelecida pelo Algorítmo 1. Isto é, após obter os mecanismos de coordenação das estrelas $\mathbf{a_j}$ do nível $\mathbf{k}$ o próximo passo do Algoritmo 1 é conectá-los com os mecanismos de coordenação das estrelas adjacentes (Definição 17) do nível $\mathbf{k-1}$.

Os mecanismos de coordenação $\mathbf{MCa_j}$ e $\mathbf{MCa_k}$ de estrelas adjacentes $\mathbf{a_j}$ e $\mathbf{a_k}$ podem ser conectados. Esta afirmação se verifica do fato da relação entre $\mathbf{a_j}$ e $\mathbf{a_k}$ ser modelada em $\mathbf{MCa_j}$ ou em $\mathbf{MCa_k}$ (Procedimento 2). Assim, uma das atividades, $\mathbf{a_j}$ ou $\mathbf{a_k}$, é comum tanto ao $\mathbf{MCa_j}$ quanto ao $\mathbf{MCa_k}$, o que satisfaz a condição de existência da operação $\oplus$ (Definição 19).

As possibilidades de conexão são determinadas combinando-se o fato da aresta definida por $\mathbf{a_j}$ e $\mathbf{a_k}$ ter rótulo unitário ou não, das atividades $\mathbf{a_j}$ e $\mathbf{a_k}$ terem ou não rótulos e de uma das atividades pertencer ou não ao rótulo da outra. Para qualquer uma destas possibilidades, a conexão $\mathbf{MCa_j \oplus MCa_k}$ pode ser efetuada aplicando-se o Procedimento 1 conforme estabelece a Proposição 4.

Proposição 4:

Se duas estrelas são adjacentes então a operação de conexão entre seus mecanismos de coordenação é efetuada aplicando-se o **Procedimento 1**.

Demonstração:

Sejam a estrela $\mathbf{a_j}$ e a estrela $\mathbf{a_k}$ estrelas adjacentes e $\mathbf{MCa_j}$ e $\mathbf{MCa_k}$ seus respectivos mecanismos de coordenação.

Por hipótese uma das atividades é representada tanto no $\mathbf{MCa_j}$ quanto no $\mathbf{MCa_k}$, logo a operação de conexão $\mathbf{MCa_j \oplus MCa_k}$ está definida.

Seja $\mathbf{a_j}$ a atividade comum à $\mathbf{MCa_j}$ e $\mathbf{MCa_k}$. Do processo de construção do mecanismo de coordenação tem-se que:

i) $\mathbf{a_j}$ possui uma transição que promove o início de sua execução e uma transição que promove o fim de sua execução ou;

ii) $\mathbf{a_j}$ possui duas transições que promovem o início de sua execução, isto é, uma transição global e uma transição local, e uma transição que promove o fim de sua execução.

De *i*) e *ii*) tem-se os seguintes casos:

c1) $\mathbf{a_j}$ está modelada da forma *i*) no $\mathbf{MCa_j}$ e *i*) no $\mathbf{MCa_k}$;

c2) $\mathbf{a}_j$ está modelada da forma *i*) no $\mathbf{MCa}_j$ e *ii*) no $\mathbf{MCa}_k$;

c3) $\mathbf{a}_j$ está modelada da forma *ii*) no $\mathbf{MCa}_j$ e *i*) no $\mathbf{MCa}_k$;

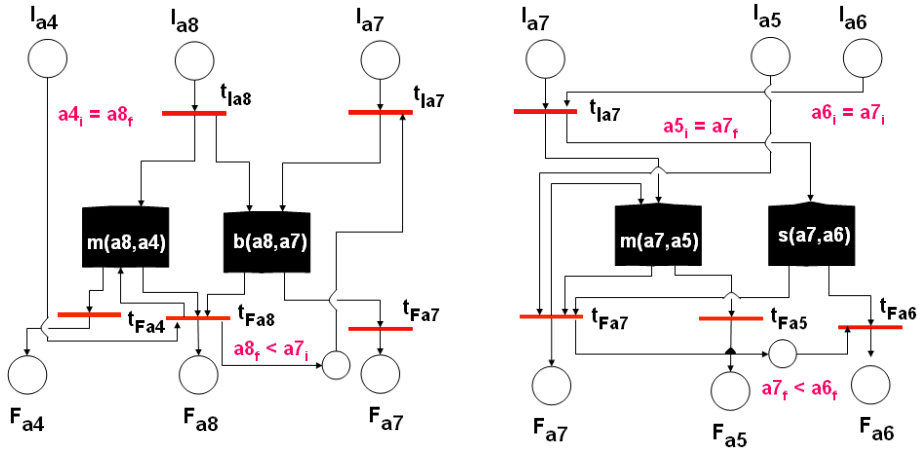
c4) $\mathbf{a}_j$ está modelada da forma *ii*) no $\mathbf{MCa}_j$ e *ii*) no $\mathbf{MCa}_k$.

Assim, considerando-se $\mathbf{MC}_1 = \mathbf{MCa}_j$ e $\mathbf{MC}_2 = \mathbf{MCa}_k$, a operação $\mathbf{MC}_1 \oplus \mathbf{MC}_2$ é efetuada da seguinte forma:

- para o caso **c1** aplica-se o item (a) do Procedimento 1;
- para os casos **c2** e **c3** aplica-se o item (b) do Procedimento 1**;** e
- para **c4** aplica-se o item (c) do Procedimento 1.

Portanto a operação $\mathbf{MCa}_j \oplus \mathbf{MCa}_k$ é efetuada aplicando-se o Procedimento 1■

A fim de exemplificar o uso da operação $\oplus$ para mecanismos derivados de estrelas considere as estrelas **a7** e **a8** da expressão **(E2,G2)** representada na Figura 3-5. O centro da estrela **a8** pertence a curva de nível $\mathbf{I}_1$ e o centro da estrela **a7** pertence a curva de nível $\mathbf{I}_2$. Assim, o Algoritmo 1 gera primeiro o mecanismo da estrela **a8** (isto é,- $\mathbf{MCa}_8$) (Figura 3-35(a)) e depois o da estrela **a7** (isto é, $\mathbf{MCa}_7$) (Figura 3-35(b)). Do **Procedimento 2** passo 3 tem-se que a relação temporal entre $\mathbf{a}_7$ e $\mathbf{a}_8$ é modelada no $\mathbf{MCa}_8$.



(a) $\mathbf{MCa}_8$ **(b)** $\mathbf{MCa}_7$

Figura 3-35: Mecanismos de coordenação das estrelas **a7** e **a8**.

Da Proposição 4 tem-se que a operação de conexão entre $\mathbf{MCa}_8$ e $\mathbf{MCa}_7$ é efetuada aplicando-se o item (a) do Procedimento 1, pois a atividade $\mathbf{a}_7$, que viabiliza a conexão, possui uma transição que promove o início de sua execução e uma transição que promove o fim de sua execução tanto em $\mathbf{MCa}_8$ como em

$\mathbf{MCa_7}$. A Figura 3 - 36 ilustra o mecanismo $\mathbf{MCa_8 \oplus MCa_7}$ resultante dessa operação de conexão.

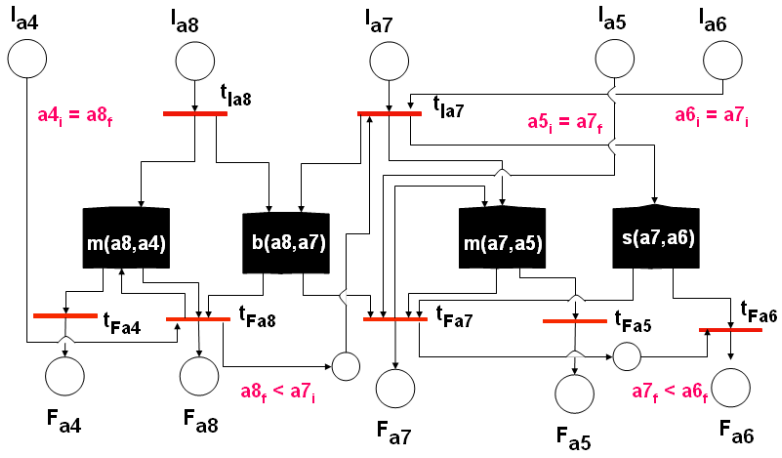


Figura 3 - 36: Operação de conexão $\mathbf{MCa_8 \oplus MCa_7}$

Finalizando esta seção apresenta-se novamente o Algoritmo 1, agora inserindo-se os refinamentos desenvolvidos ao longo deste capítulo

Algoritmo 1: Identificação e modelagem das **CGs**

Dada uma expressão ε

Obter Partição de **A** : $\mathbf{P(A) = \{I_0, I_1, \cdots, I_m\}}$

Para $\mathbf{k \leftarrow 1}$ passo **1** até (número de elementos de $\mathbf{P(A)}$) faça

passo 1. selecionar o conjunto $\mathbf{I_k}$ de atividades $\mathbf{a_i}$;

passo 2. identificar e modelar a lista de **restrições diretas** para cada estrela $\mathbf{a_i}$, $\mathbf{a_i \in I_k}$.

passo 3. conectar os Mecanismos de Coordenação das estrelas $\mathbf{a_i's}$, $\mathbf{a_i \in I_k}$ com os Mecanismos de Coordenação das estrelas $\mathbf{a_j's}$, $\mathbf{a_j \in I_{k-1}}$ apropriados.

Fim para

Se o centro[14] $\mathbf{I_m}$ de ε tiver duas atividades $\mathbf{a_p}$ e $\mathbf{a_q}$

então conectar $\mathbf{MCa_p}$ com o $\mathbf{MCa_q}$;

Fim algoritmo

Algoritmo 1: Identificação e modelagem das Condições Globais

[14] O centro da expressão $\mathbf{I_m}$ sempre terá uma ou duas atividades conforme estabelecido na Seção **3.3.2**

Complexidade do Algoritmo 1

Demonstra-se a seguir que o Algoritmo 1 proposto para construção do mecanismo de coordenação de uma expressão ε com **n** atividades é O(n).
Teorema 2

O Algoritmo 1 tem complexidade O(n).

Demonstração

A demonstração do Teorema 2 é feita através da seqüência de fatos apresentados a seguir.

Para selecionar o conjunto de atividades $\mathbf{I_k}$, $\mathbf{k = 1,2,\cdots, m < \frac{n}{2}}$, da k-ésima iteração é necessário identificar todas as atividades de grau 1, o que requer **O(n)** operações para cada iteração. Logo o algoritmo torna-se $O(n^2)$. O que se faz para manter a linearidade é calcular o grau de todas as atividades uma única vez e a cada iteração a relação entre o grau das atividades é atualizado com a remoção das folhas de ε_k (Definição 15). Isto evita a necessidade de calcular o grau das atividades para cada nova iteração. Assim fica apenas o custo da operação de remoção de atividades para ser efetuada a cada iteração, o que é $O(n)$, pois existem **n** atividades e que são removidas uma única vez.

O custo do processo de construção do **MCG** para uma estrela é dado em função do seu número de relacionamentos, sendo $O(1)$ para cada relação. Uma relação pertence ao mecanismo de coordenação de uma única estrela. O número máximo de relações é $\mathbf{7(n-1)}$, considerando o caso em que cada uma das $\mathbf{n-1}$ arestas tem rótulo com 7 elementos. Logo a construção do **MCG** para todas as estrelas de expressão é $O(n)$.

Por fim, o custo para conectar os mecanismos de coordenação das estrelas também é $O(n)$, pois o número de estrelas sempre é menor do que o número de atividades da expressão.

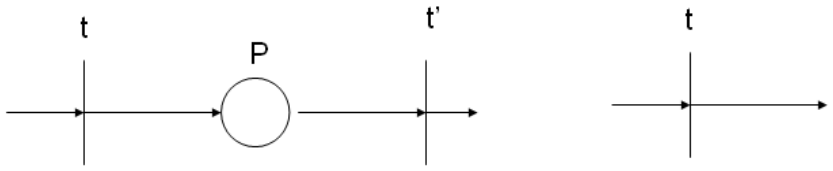
Portanto o Algoritmo 1 é linear em relação ao número de atividades■.

3.4. Consistência dos Mecanismos de Coordenação

Um mecanismo de coordenação, obtido através do Algoritmo 1, é o resultado da modelagem de todas as restrições temporais decorrentes dos relacionamentos entre as atividades da aplicação. Este mecanismo pode ser usado na coordenação de uma

aplicação somente se ele for consistente, isto é, todas as restrições temporais modeladas por ele podem ser satisfeitas.

As restrições temporais são modeladas (Seção 3.3.3.2) conforme os diagramas da Figura 3-37. Diz-se que uma restrição temporal envolvendo uma inequação (Figura 3-37(a)) ou uma equação (Figura 3-37(b)) é satisfeita quando ocorre o disparo da transição **t** envolvida na modelagem da referida restrição.



(a) inequação **(b)** equação

Figura 3-37: Modelagem de uma restrição temporal

Definição 22: Mecanismo de Coordenação consistente

> Um mecanismo de coordenação é consistente se e somente se todas as restrições temporais modeladas podem ser satisfeitas.

Com a finalidade de provar a consistência de um mecanismo de coordenação, isto é, de um **MC**, mostra-se a seguir que os **MCs** básicos são consistentes.

Os **MCs** básicos são aqueles que não podem ser obtidos através de operações de conexão. Existem nove **MCs** básicos na metodologia **GR**: os sete **MCs** correspondentes as relações do conjunto **D** (Definição 6), o **MC** correspondente ao rótulo de aresta e o **MC** correspondente ao rótulo de atividade.

Teorema 3

> Um mecanismo de coordenação básico é consistente.

Demonstração

Apresenta-se a seguir uma prova por exaustão.

i. O **MC** da relação "before" $\mathbf{b}(\mathbf{a_j}, \mathbf{a_k})$, representado pelo diagrama da Figura 3-19(a), é consistente. Isto se verifica, pois uma solicitação para executar a atividade $\mathbf{a_j}$, que equivale a uma ficha no lugar $\mathbf{Ia_j}$, satisfaz a restrição temporal $(\mathbf{aj_f} < \mathbf{ak_i})$.

ii. Para o **MC** da relação "during" $\mathbf{d(a_j, a_k)}$, representado pelo diagrama da Figura 3-19 (b), tem-se que: uma solicitação para executar a atividade $\mathbf{a_k}$ (isto é, uma ficha no lugar $\mathbf{Ia_k}$) satisfaz a restrição temporal $\mathbf{(aj_i > ak_i)}$. Somando à isto, uma solicitação para executar a atividade $\mathbf{a_j}$ (isto é, uma ficha no lugar $\mathbf{Ia_j}$) satisfaz a restrição temporal $\mathbf{(aj_f < ak_f)}$. Logo este **MC** é consistente.

iii. De forma análoga aos itens i e ii tem-se que os **MCs** das relações "overlaps", "starts", "finishes", "equal" e "meets" entre as atividades $\mathbf{a_j}$ e $\mathbf{a_k}$, representados respectivamente pelos diagramas das Figura 3-20, Figura 3-21(a), Figura 3-21(b), Figura 3-22 (a) e Figura 3-22 (b), são consistentes. Isto se verifica porque uma solicitação para executar a atividade $\mathbf{a_j}$ e outra para executar a atividade $\mathbf{a_k}$ satisfazem as restrições temporais modeladas pelos referidos **MCs**. Portanto estes **MCs** são consistentes.

iv. O **MC** do rótulo $\mathbf{\{r_1, r_2, \cdots, r_q\}}$ da aresta $\mathbf{(a_j, a_k)}$ (Figura 3-23) é consistente se satisfizer as restrições temporais correspondentes a qualquer uma das relações do seu rótulo. Para que ocorra a seleção da relação $\mathbf{r_i \in \{r_1, r_2, \cdots, r_q\}}$ é necessária uma solicitação para executar a atividade $\mathbf{a_j}$ e uma solicitação para executar a atividade $\mathbf{a_k}$, isto é, uma ficha de cor $\mathbf{< r_i, a_j, a_k >}$ no lugar $\mathbf{Ia_j}$ e outra ficha de mesma cor no lugar $\mathbf{Ia_k}$. Considerando que a modelagem de $\mathbf{r_i}$ corresponde a um **MC'** de uma das relações do conjunto **D** (Definição 6) e as solicitações para execução de $\mathbf{a_j}$ e $\mathbf{a_k}$ tem-se que as restrições temporais derivadas de $\mathbf{r_i}$ também são satisfeitas. Portanto o **MC** do rótulo $\mathbf{\{r_1, r_2, \cdots, r_q\}}$ da aresta $\mathbf{(a_j, a_k)}$ é consistente.

v. De forma análoga ao item iv tem-se que o **MC** do rótulo $\mathbf{\{b_1, b_2, \cdots, b_q\}}$da atividade $\mathbf{a_j}$ (Figura 3-24) é consistente. Isto é, uma solicitação para executar a atividade $\mathbf{a_j}$ e uma solicitação para executar a atividade $\mathbf{b_i}$, $\mathbf{i \in \{1,2,\cdots,q\}}$, que equivale a uma ficha de cor $\mathbf{< r_i, a_j, b_i >}$ no lugar $\mathbf{Ia_j}$ e outra ficha de mesma cor no lugar $\mathbf{Ib_i}$ promove a seleção da relação $\mathbf{r_i}$ entre as atividades $\mathbf{a_j}$ e $\mathbf{b_i}$. Considerando que a modelagem de $\mathbf{r_i}$ corresponde ao mecanismo de coordenação de uma das relações do conjunto **D** (Definição 6) e as solicitações para execução de $\mathbf{a_j}$ e $\mathbf{b_i}$ tem-se que as restrições temporais derivadas de $\mathbf{r_i}$ também são satisfeitas. Logo o **MC** do rótulo $\mathbf{\{b_1, b_2, \cdots, b_q\}}$ da atividade $\mathbf{a_j}$ é consistente.

Finalmente, dos itens i, ii, iii, iv e v conclui-se que um **MC** básico é consistente.

Para provar a consistência de um mecanismo de coordenação deve-se mostrar que a conexão de **MCs** consistentes é um **MC** consistente. Este resultado é estabelecido pelo Teorema 4 apresentado a seguir.

Teorema 4

A conexão de mecanismos de coordenação consistentes é um mecanismo de coordenação consistente.

Demonstração

Sejam $\mathbf{MC_1}$ e $\mathbf{MC_2}$ dois **MCs** básicos e $\mathbf{MC' = MC_1 \oplus MC_2}$ o mecanismo obtido efetuando-se a operação de conexão entre $\mathbf{MC_1}$ e $\mathbf{MC_2}$. Pelo Teorema 3 tem-se que $\mathbf{MC_1}$ e $\mathbf{MC_2}$ são consistentes. Como a operação de conexão não insere nem remove uma restrição temporal tem-se que **MC'** é a união das restrições modeladas em $\mathbf{MC_1}$ e $\mathbf{MC_2}$ e, portanto continuam sendo satisfeitas. Logo **MC'** é um mecanismo consistente.

Suponha, por hipótese de indução, que o mecanismo de coordenação $\mathbf{MC'' = MC_1 \oplus MC_2 \oplus \cdots \oplus MC_n}$, obtido conectando-se **n** mecanismos de coordenação básicos, é consistente.

Obtém-se agora o $\mathbf{MC = MC'' \oplus MC_{n+1}}$ efetuando-se uma operação de conexão entre **MC''** e um mecanismo de coordenação básico $\mathbf{MC_{n+1}}$. Novamente, como a operação de conexão não insere nem remove uma restrição temporal tem-se que o **MC** satisfaz todas as restrições tempoais e, portanto é um mecanismo consistente. Fica assim demonstrado o Teorama 3■.

3.5. Considerações do Capítulo

As transições com reserva de fichas permitem simular intervalos de tempo com duração não determinada. Para isto, deve-se retirar as fichas dos lugares de entrada quando a transição for habilitada e enviá-las para os lugares de saída após o evento de fim de execução da atividade associada à transição.

Com a simulação de intervalos de tempo com duração não determinada o tempo de execução de uma atividade passa a ser determinado pelo ambiente computacional em que ela é executada. Esta dinâmica permite associar diferentes tipos de eventos do ambiente computacional às transições com reserva de fichas. Por exemplo, a intervenção do usuário determinando o fim de uma ou mais atividades ou uma coleção de condições, associadas ao ambiente, determinando o fim de execução de atividades.

O próximo capítulo apresenta estudos de casos que exemplificam o uso da metodologia **GR** considerando tanto intervalos de tempo com duração determinada quanto os de duração não determinada.

4. ESTUDOS DE CASOS

O propósito deste capítulo é o de discutir os mecanismos de coordenação propostos pela metodologia **GR** através de:

- exemplos de entendimento das estruturas,
- exemplos de uso da metodologia proposta.

Estes exemplos buscam também evidenciar as capacidades da metodologia **GR** de responder às necessidades da coordenação, isto é, às necessidades de superar ou evitar os conflitos gerados pelas dependências entre as atividades.

São apresentados três estudos de caso:

- **caso 1**: uso da operação de seleção,
- **caso 2**: ambiente multimídia,
- **caso 3**: coordenação do tráfego em um sinaleiro,

para cada caso mostra-se, analítica e graficamente, a expressão **GR** que especifica o ambiente a ser coordenado e a construção do seu Mecanismo de Coordenação seguindo os passos do Algoritmo 1 (Seção 3.3).

4.1. Estudo de Caso 1 - uso da operação de seleção

A possibilidade de escolher uma atividade em um conjunto de atividades ou uma relação em um conjunto de relações, bem como uma combinação destas, confere à metodologia **GR** a capacidade de atender aplicações que necessitam de comportamentos alternativos. A seleção de atividades permite que comportamentos temporais equivalentes (relações entre as atividades) sejam definidos para diferentes atividades enquanto a seleção de relações permite especificar distintos comportamentos temporais para um mesmo conjunto de atividades.

Neste estudo de caso pretende-se evidenciar o uso da operação de seleção de atividades. Buscou-se combinar a eficiência de um exemplo simples com uma situação real de forma a esclarecer o uso desta operação.

O problema que será especificado é o de montar, conforme uma ordem pré-estabelecida, um objeto a partir de uma coleção de peças. A expressão qualificada $(\mathbf{E}(\mathbf{A},\mathbf{R},\mathbf{F}),\mathbf{G})$ apresentada a seguir estabelece os relacionamentos entre as

atividades que correspondem aos passos da montagem de uma mesa. Assim a expressão $(\mathbf{E}(\mathbf{A},\mathbf{R},\mathbf{F}),\mathbf{G})$ ou $(\mathbf{E},\mathbf{G})$ é dada por:

$\mathbf{A} = \{\mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3, \mathbf{a}_4, \mathbf{a}_5, \mathbf{a}_5, \mathbf{a}_6\}$,

$\mathbf{R} = \{(\mathbf{a}_1, \mathbf{a}_3),\ (\mathbf{a}_2, \mathbf{a}_3),\ (\mathbf{a}_3, \mathbf{a}_4), (\mathbf{a}_5, \mathbf{a}_4),\ (\mathbf{a}_6, \mathbf{a}_4)\}$,

F a função rotuladora de arestas dada por:

$\mathbf{Y}_{1,3} = \{\mathbf{b}\},\ \mathbf{Y}_{2,3} = \{\mathbf{b}\}, \mathbf{Y}_{3,4} = \{\mathbf{b}\}, \mathbf{Y}_{5,4} = \{\mathbf{b}\}$ e $\mathbf{Y}_{6,4} = \{\mathbf{b}\}$

G a função rotuladora de atividades dada por:

$\mathbf{X}_3 = \{\mathbf{a}_1, \mathbf{a}_2\}$,

$\mathbf{X}_4 = \{\mathbf{a}_5, \mathbf{a}_6\}$, e

$\mathbf{X}_1 = \mathbf{X}_2 = \mathbf{X}_5 = \mathbf{X}_6 = \{\varnothing\}$.

As atividades em **A** correspondem a:

Atividade	descrição
a1	Gerar pernas tipo 1
a2	Gerar pernas tipo 2
a3	Gerar quadro e conectar com a(s) perna(s) da mesa
a4	Conectar tampo ao restante da mesa
a5	Gerar tampo da mesa tipo 1
a6	Gerar tampo da mesa tipo 2

Devido às possibilidades de relacionamentos das atividades $\mathbf{a}_3$ (com $\mathbf{a}_1$ e $\mathbf{a}_2$) e $\mathbf{a}_4$(com $\mathbf{a}_5$ e $\mathbf{a}_6$) verifica-se que ao todo pode-se montar 4 mesas diferentes. A Figura 4 - 1 ilustra a representação gráfica da expressão $(\mathbf{E},\mathbf{G})$.

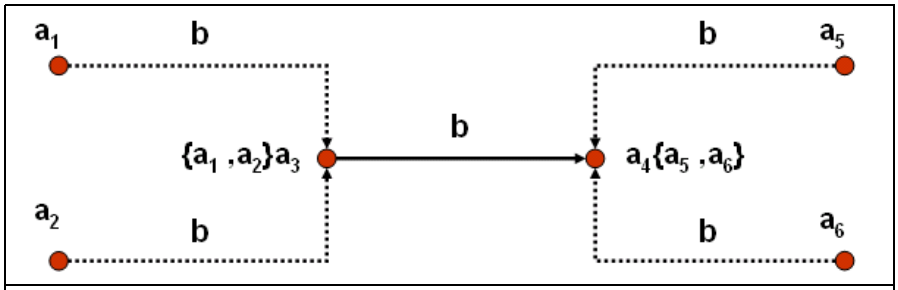


Figura 4 - 1 : Construção de mesa $(\mathbf{E},\mathbf{G})$.

Seguindo o Algoritmo1 para construção do **MC** (identificação e modelagem das restrições temporais) deve-se selecionar (Seção 3.3.2) uma curva de nível a cada

iteração começando por $\mathbf{I_1}$. $(\mathbf{E,G})$ tem apenas duas curvas de nível: $\mathbf{I_0} = \{\mathbf{a_1, a_2, a_5, a_6}\}$ e $\mathbf{I_1} = \{\mathbf{a_3, a_4}\}$; selecione-se a curva de nível $\mathbf{I_1}$.

No próximo passo, deve-se determinar os mecanismos de coordenação das estrelas com centro em $\mathbf{I_1}$, isto é, $\mathbf{MCa_3}$ e $\mathbf{MCa_4}$. Aplicando-se o **Procedimento 2** (Seção 3.3.3.2.) para cada estrela tem-se que:

$\mathbf{MCa_3} = \mathbf{MC_1} \oplus \mathbf{MC_2}$ e $\mathbf{MCa_4} = \mathbf{MC_3}$, onde

$\mathbf{MC_1}$ é o mecanismo correspondente ao rótulo de $\mathbf{a_3}$;
$\mathbf{MC_2}$ é o mecanismo correspondente ao rótulo $\mathbf{Y_{3,4}} = \{\mathbf{b}\}$, e
$\mathbf{MC_3}$ é o mecanismo correspondente ao rótulo de $\mathbf{a_4}$.

O $\mathbf{MC_1}$ é a modelagem das restrições diretas da estrela $\mathbf{a_3}$ determinadas pelo conjunto $\mathbf{C_2}$ (Definição 16), formado por disjunção entre o conjunto de restrições derivados do rótulo $\mathbf{Y_{1,3}}$ e o conjunto de restrições derivados do rótulo $\mathbf{Y_{2,3}}$:

$$\mathbf{C_2} = \{\mathbf{Y_{1,3}}\ \mathbf{.ou.}\ \mathbf{Y_{2,3}}\} = \{\{\mathbf{b}\}\ \mathbf{.ou.}\ \{\mathbf{b}\}\} =$$
$$= \{\ (\mathbf{a_{1f} < a_{3i}})\ \mathbf{.ou.}\ (\mathbf{a_{2f} < a_{3i}})\ \}$$

A fim de obter-se $\mathbf{MC_1}$, observa-se que das atividades coordenadas por ele ($\mathbf{a_1}$, $\mathbf{a_2}$ e $\mathbf{a_3}$), $\mathbf{a_3}$ é a única cujo grau é maior do que um, $\partial(\mathbf{a_3}) = \mathbf{2}$. Ou seja, a conexão de $\mathbf{MC_1}$ com outro mecanismo, no caso $\mathbf{MC_2}$, acontece através de $\mathbf{a_3}$. Assim, $\mathbf{MC_1}$ deve prover as funcionalidades de conexão ("dispositivos" que permitem selecionar e transmitir para outros **MCs** as restrições temporais impostas à $\mathbf{a_3}$) para esta atividade.

Por questão de clareza na apresentação do processo de modelagem, considera-se inicialmente $\mathbf{MC_1}$ sem estas funcionalidades. Dessa forma, utilizando-se o diagrama **CPNet 4** (Figura 3-24) obtém-se, uma versão ainda incompleta, $\mathbf{MC_1}$ conforme ilustra a Figura 4-2.

Os retângulos pontilhados da Figura 4-2 indicam os mecanismos que modelam as relações entre $\mathbf{a_3}$ e as atividades do seu rótulo($\mathbf{X_3} = \{\mathbf{a_1, a_2}\}$). Estes mecanismos estão inicialmente estruturados de forma idêntica ao diagrama **CPNet 2** (Figura 3-18), para que eles caracterizem apropriadamente a relação **before** entre $\mathbf{a_3}$ e $\mathbf{a_1}$ e a relação **before** entre $\mathbf{a_3}$ e $\mathbf{a_2}$ deve-se adicionar-lhes as restrições temporais conforme o diagrama da Figura 3-19(a). A Figura 4-3 apresenta o $\mathbf{MC_1}$ com estas alterações. Ainda nesta figura, substitui-se as "caixas pretas" (representação do **MCL**) da Figura 4-2 pelos seus respectivos mecanismos de coordenação local (Figura 3-16).

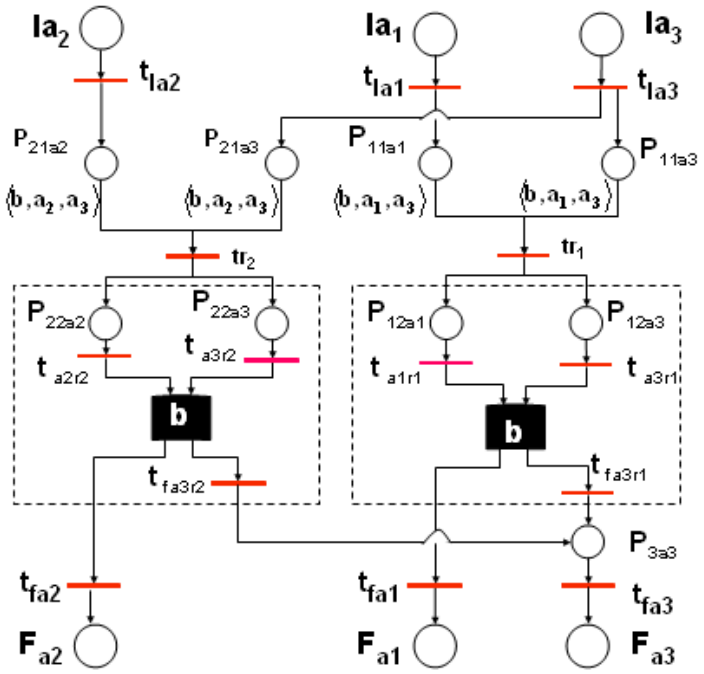


Figura 4 - 2: $\mathbf{MC_1}$ mecanismo correspondente ao rótulo de $\mathbf{a_3}$ (incompleto-v1).

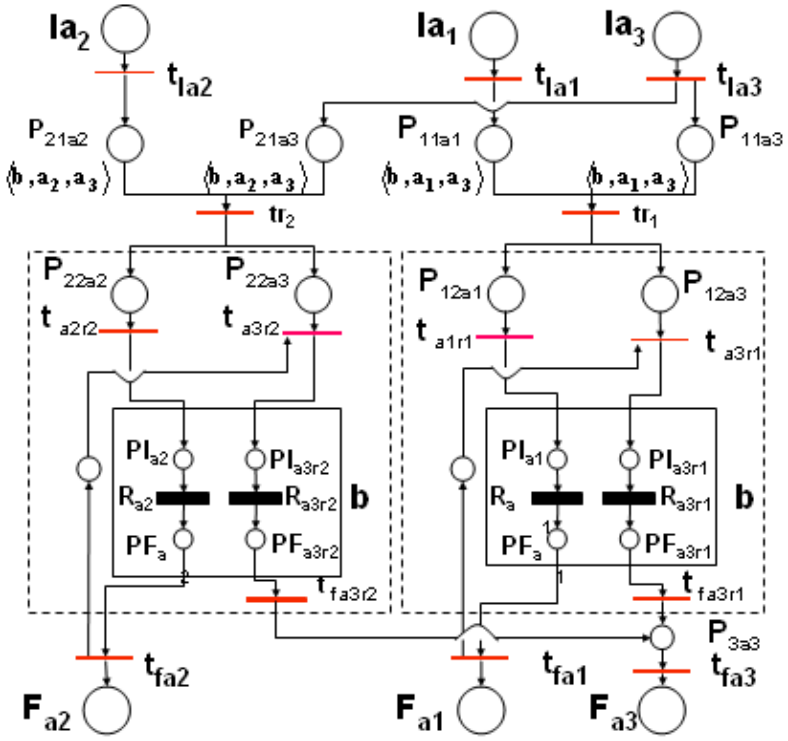


Figura 4 - 3: $\mathbf{MC_1}$ mecanismo correspondente ao rótulo de $\mathbf{a_3}$ (incompleto-v2).

Por fim, adiciona-se o dispositivo de conexão para $\mathbf{a}_3$ conforme o diagrama Figura 3 - 33 . Para tal, procede-se conforme os itens a seguir:

1. alterar o nome da transição $\mathbf{t}_{\mathbf{Ia3}}$ para $\mathbf{t}_{\mathbf{1Ia3}}$ e adicionar a transição $\mathbf{t}_{\mathbf{2Ia3}}$
2. alterar o nome da transição $\mathbf{t}_{\mathbf{a3r1}}$ para $\mathbf{t}_{\mathbf{1a3r1}}$ e adicionar a transição $\mathbf{t}_{\mathbf{2a3r1}}$
3. alterar o nome da transição $\mathbf{t}_{\mathbf{a3r2}}$ para $\mathbf{t}_{\mathbf{1a3r2}}$ e adicionar a transição $\mathbf{t}_{\mathbf{2a3r2}}$
4. adicionar os lugares $\mathbf{P}_{\mathbf{1a3}}$ e $\mathbf{P}_{\mathbf{2a3}}$.

A Figura 4 - 4 apresenta a versão final para $\mathbf{MC}_1$.

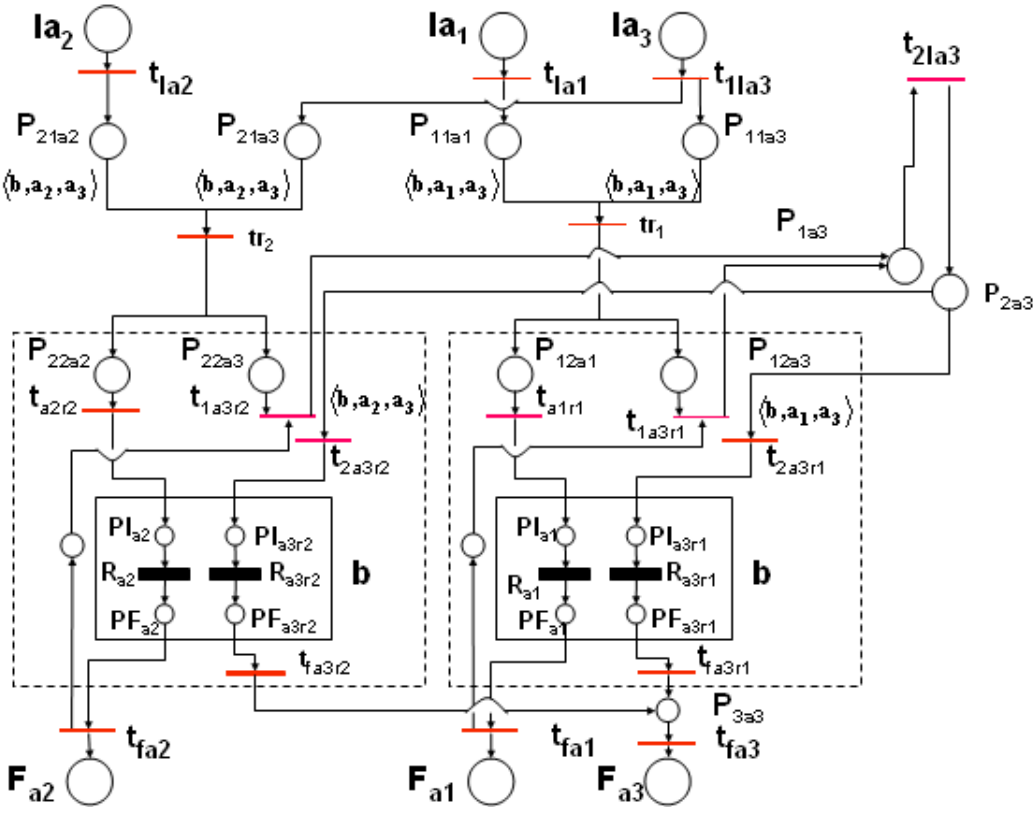


Figura 4 - 4 : $\mathbf{MC}_1$ mecanismo correspondente ao rótulo de $\mathbf{a}_3$.

Como dito anteriormente, as atividades $\mathbf{a}_1$ e $\mathbf{a}_2$ são atividades folha, $\partial(\mathbf{a}_1)=\partial(\mathbf{a}_2)=\mathbf{1}$, e portanto não é necessário modelar-se os elementos de conexão para cada uma delas conforme feito para $\mathbf{a}_3$.

O $\mathbf{MC}_2$ é a modelagem das restrições diretas da estrela $\mathbf{a}_3$ determinadas pelo conjunto $\mathbf{C}_1$ conforme estabelece a Definição 16:

$$\mathbf{C}_1=\{\mathbf{Y}_{3,4}\}=\{\{\mathbf{b}\}\}=\{(\mathbf{a}_{3f}<\mathbf{a}_{4i})\}$$

Do diagrama apresentado na Figura 3-19 (a) (**before**) deriva-se diretamente o mecanismo de coordenação $\mathbf{MC}_2$ da relação entre $\mathbf{a}_3$ e $\mathbf{a}_4$, conforme ilustra a Figura 4-5.

Por fim, efetuando-se a operação de conexão entre $\mathbf{MC}_1$ (Figura 4-4) e $\mathbf{MC}_2$ (Figura 4-5) obtém-se o mecanismo $\mathbf{MCa}_3$, responsável por coordenação as restrições temporais da estrela $\mathbf{a}_3$, isto é, $\mathbf{La}_3 = \mathbf{C}_1 \mathbf{.e.} \mathbf{C}_2$. A operação $\mathbf{MC}_1 \oplus \mathbf{MC}_2$ pode ser efetuada porque a atividade $\mathbf{a}_3$ é comum aos dois mecanismos.

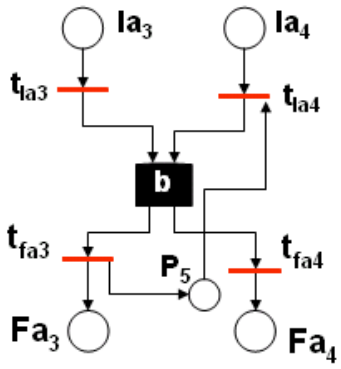


Figura 4-5: $\mathbf{MC}_2$ mecanismo correspondente a relação $\mathbf{b}(\mathbf{a}_3, \mathbf{a}_4)$.

Para tal tem-se que:

- o início de $\mathbf{a}_3$, em $\mathbf{MC}_1$, está associado a duas transições, $\mathbf{t}_{1Ia3}$ e $\mathbf{t}_{2Ia3}$, e o fim à transição $\mathbf{t}_{fa3}$;
- o início de $\mathbf{a}_3$, em $\mathbf{MC}_2$, está associado a transição $\mathbf{t}_{Ia3}$ e o fim à transição $\mathbf{t}_{fa3}$.

Esta configuração determina que a operação $\mathbf{MC}_1 \oplus \mathbf{MC}_2$ é efetuada aplicando-se o item **(b)** do Procedimento 1. Esta configuração sempre ocorre ao conectar-se o mecanismo derivado do rótulo de uma atividade com um mecanismo derivado de uma única relação.

Portanto, $\mathbf{MC}_1 \oplus \mathbf{MC}_2$ é o resultado da fissão (Definição 21) da transição $\mathbf{t}_{Ia3}$ de $\mathbf{MC}_2$ nas transições $\mathbf{t}_{1Ia3}$ e $\mathbf{t}_{2Ia3}$ de $\mathbf{MC}_1$ e da fusão (Definição 20) da transição $\mathbf{t}_{fa3}$ de $\mathbf{MC}_2$ com a transição $\mathbf{t}_{fa3}$ de $\mathbf{MC}_1$ gerando $\mathbf{t}_{fa3}$ na Figura 4-6.

A Figura 4-6 apresenta o mecanismo $\mathbf{MCa}_3$ resultante de $\mathbf{MC}_1 \oplus \mathbf{MC}_2$, que corresponde à modelagem das restrições diretas da estrela $\mathbf{a}_3$ determinadas por $\mathbf{La}_3 = \{\mathbf{C}_1 \mathbf{.e.} \mathbf{C}_2\}$. A partir deste ponto do texto os **MCLs** voltam a ser representados por "caixas pretas".

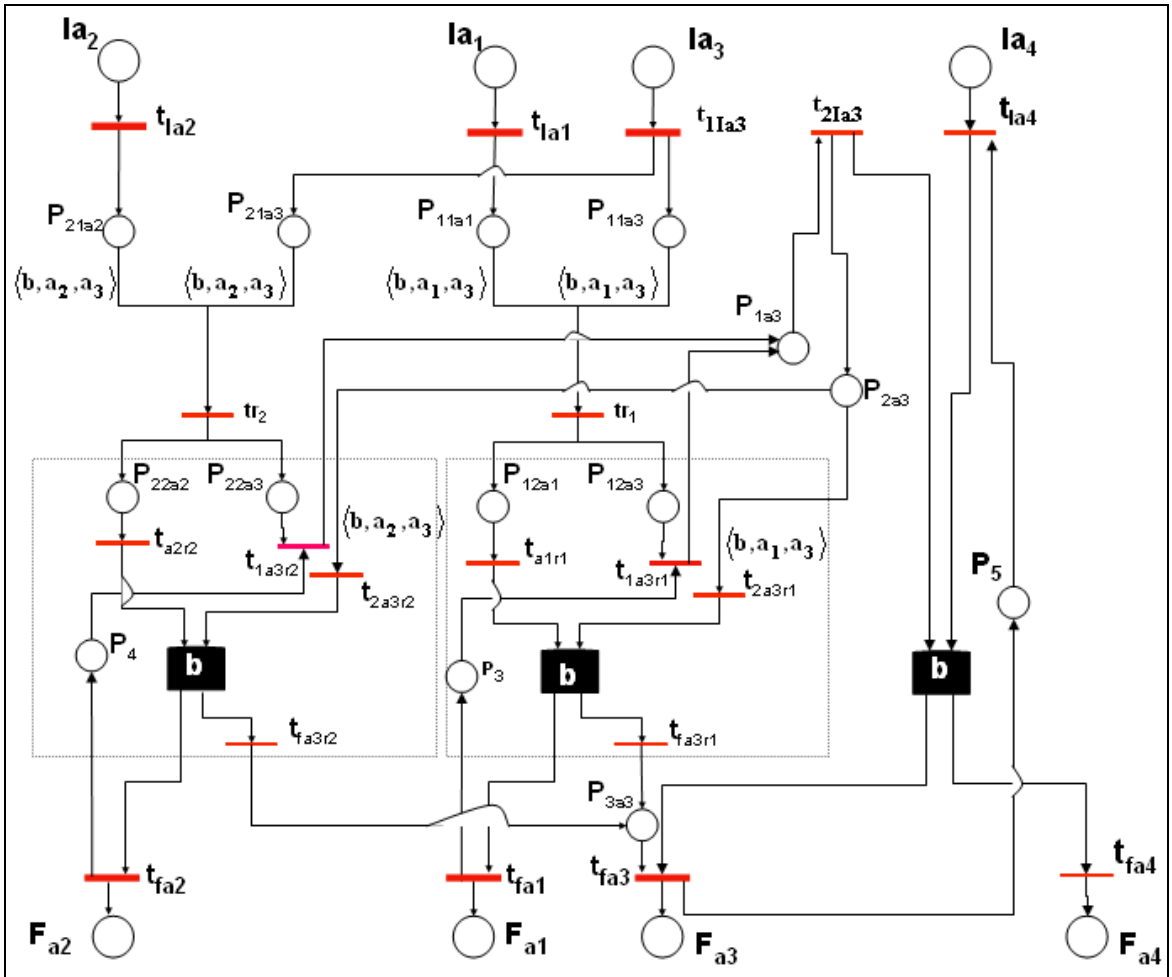


Figura 4 - 6: Mecanismo de coordenação da estrela $\mathbf{a_3}$ - $\mathbf{MCa_3}$.

Obtido o mecanismo de coordenação da estrela $\mathbf{a_3}$ passa-se a construir o mecanismo que irá coordenar a lista de restrições diretas da estrela $\mathbf{a_4}$ (Definição 16), isto é, $\mathbf{La_4 = C_1 \ .e. \ C_2}$, onde

$$\mathbf{C_1 = \{Y_{3,4}\} = \{\{b\}\} = \{(a_{3f} < a_{4i})\}}$$

$$\mathbf{C_2 = \{Y_{5,4} \ .ou. \ Y_{6,4}\} = \{\{b\} .ou. \{b\}\} =}$$
$$\mathbf{= \{ \ (a_{5f} < a_{4i}) \ .ou. \ (a_{6f} < a_{4i}) \ \}}$$

Do Procedimento 2 sabe-se que esse mecanismo coordena apenas as relações do rótulo de $\mathbf{a_4}$, isto é, $\mathbf{MCa_4 = MC_3}$, onde $\mathbf{MC_3}$ é o mecanismo correspondente ao rótulo de $\mathbf{a_4}$, como dito anteriormente.

O $\mathbf{MC}_3$ envolvendo as atividades $\mathbf{a}_4$, $\mathbf{a}_5$ e $\mathbf{a}_6$ é obtido do diagrama **CPNet 4** (Figura 3 - 33) adicionando-se o dispositivo de conexão para $\mathbf{a}_4$ ($\partial(\mathbf{a}_4)=\mathbf{2}$) conforme ilustra a Figura 4 - 7 .

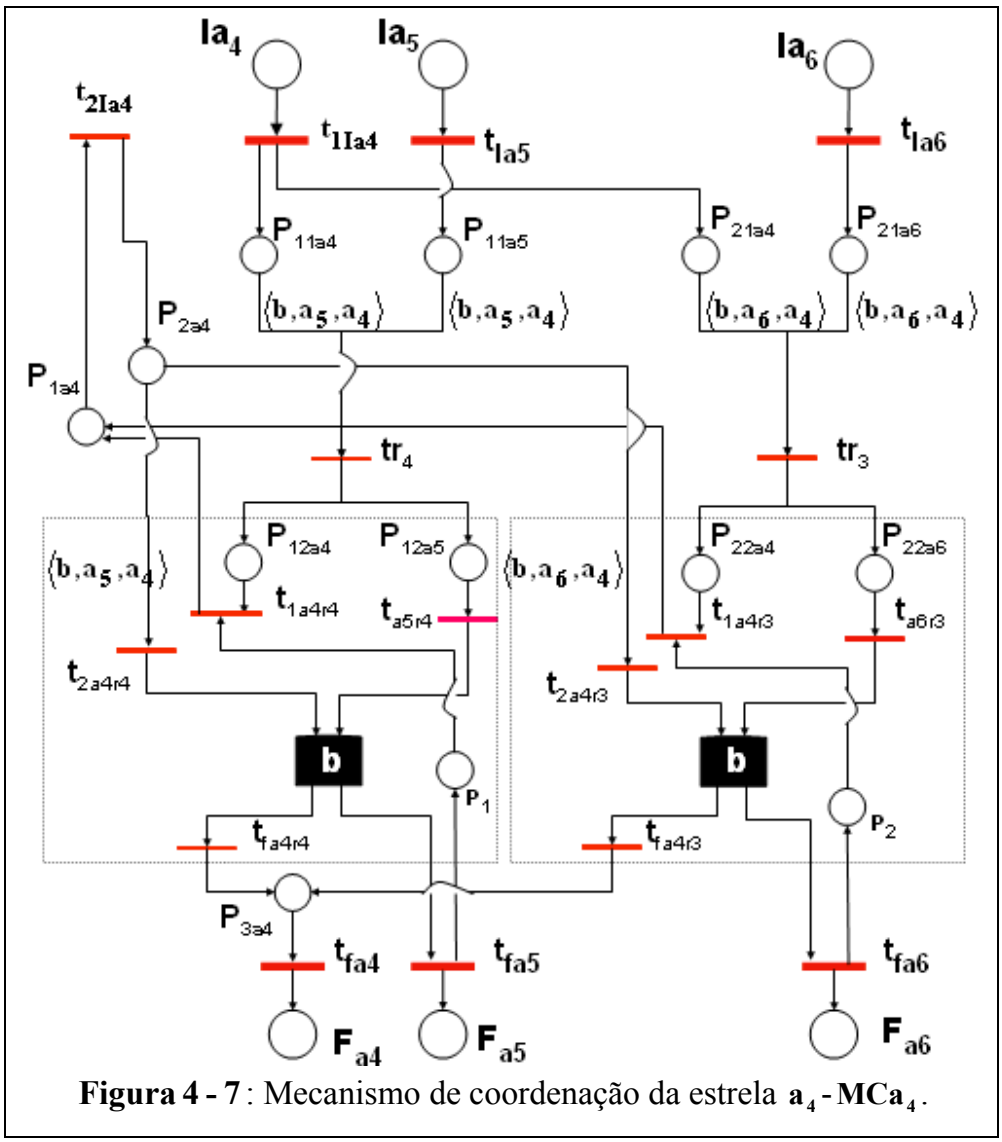


Figura 4 - 7: Mecanismo de coordenação da estrela $\mathbf{a}_4$ - $\mathbf{MCa}_4$.

O próximo passo é conectar os mecanismos de coordenação das estrelas da curva de nível anterior (no caso $\mathbf{I}_0$) com os mecanismos das estrelas geradas no nível atual (no caso $\mathbf{I}_1$). Para as estrelas da curva de nível $\mathbf{I}_0$ o Algoritmo 1 efetua estas operações ao mesmo tempo que gera os mecanismos de coordenação das estrelas de $\mathbf{I}_1$. Como $\mathbf{I}_1$ é o centro (Seção 3.3.2.) da expressão com duas atividades: $\mathbf{a}_3$ e $\mathbf{a}_4$, resta conectar os mecanismos $\mathbf{MCa}_3$ e $\mathbf{MCa}_4$ derivados das respectivas estrelas(Algoritmo 1).

As estrelas $\mathbf{a}_3$ e $\mathbf{a}_4$ são adjacentes (Definição 17), logo existe uma atividade comum a $\mathbf{MCa}_3$ e $\mathbf{MCa}_4$, no caso $\mathbf{a}_4$, que oferece as condições necessárias para efetuar a operação de conexão $\mathbf{MCa}_3 \oplus \mathbf{MCa}_4$ (veja na Figura 4-6 as transições $\mathbf{t}_{Ia4}$ e $\mathbf{t}_{fa4}$ e na Figura 4-7 as transições $\mathbf{t}_{1Ia4}$, $\mathbf{t}_{2Ia4}$ e $\mathbf{t}_{fa4}$ criadas através do **Algoritmo 1** passo2).

De acordo com a Proposição 4 a operação $\mathbf{MCa}_3 \oplus \mathbf{MCa}_4$ é efetuada pelo item **(b)** do Procedimento 1. O mecanismo resultante **MC**, apresentado na Figura 4-8, é obtido efetuando-se a fissão da transição $\mathbf{t}_{Ia4}$ de $\mathbf{MCa}_3$ nas transições $\mathbf{t}_{1Ia4}$ e $\mathbf{t}_{2Ia4}$ de $\mathbf{MCa}_4$ e uma fusão da transição $\mathbf{t}_{fa4}$ de $\mathbf{MCa}_3$ com a transição $\mathbf{t}_{fa4}$ de $\mathbf{MCa}_4$. **MC** corresponde ao Mecanismo de Coordenação da expressão $(\mathbf{E}, \mathbf{G})$.

Com este estudo de caso buscou-se exemplificar o uso da estrutura de seleção mostrando a sua capacidade de modelar comportamentos alternativos. Esta capacidade da metodologia **GR** pode ser explorada em situações que necessitem assumir diferentes comportamentos em resposta a diferentes ocorrências no ambiente computacional. Por exemplo, em sistemas de animação comportamental deseja-se modelar os personagens de modo a interagirem autonomamente em seu mundo virtual. Estes personagens devem ter a capacidade de perceber situações do ambiente e em função destas, selecionar comportamentos adequados para atuarem no mesmo [Millar 99]. Este mapeamento percepção - ação pode ser especificado usando a metodologia **GR** com a vantagem de gerar-se automaticamente o mecanismo de coordenação para os comportamentos do personagem.

Os participantes[15] de Ambientes Virtuais Colaborativos - CVE podem interagir entre si ou com objetos do mundo virtual a fim de efetuarem atividades relacionadas a um trabalho cooperativo. Novamente, a metodologia **GR** pode ser usada no processo de escolha destes objetos ou participantes e na coordenação do conjunto de atividades envolvidas no trabalho.

De forma geral, selecionar um ou outro comportamento (alternativa) é uma necessidade básica quando se especifica a solução de um problema. Além disto, a carência de uma metodologia para geração automática de mecanismos de coordenação com esta funcionalidade, conforme a literatura consultada, foram fatores motivantes para adicioná-la a metodologia **GR**.

[15] Participantes de sistemas CVE são representados no ambiente por objetos gráficos, denominados "avatares", que informam suas identidades, localização, presença e atividades com outros "avatares" [Benford 01].

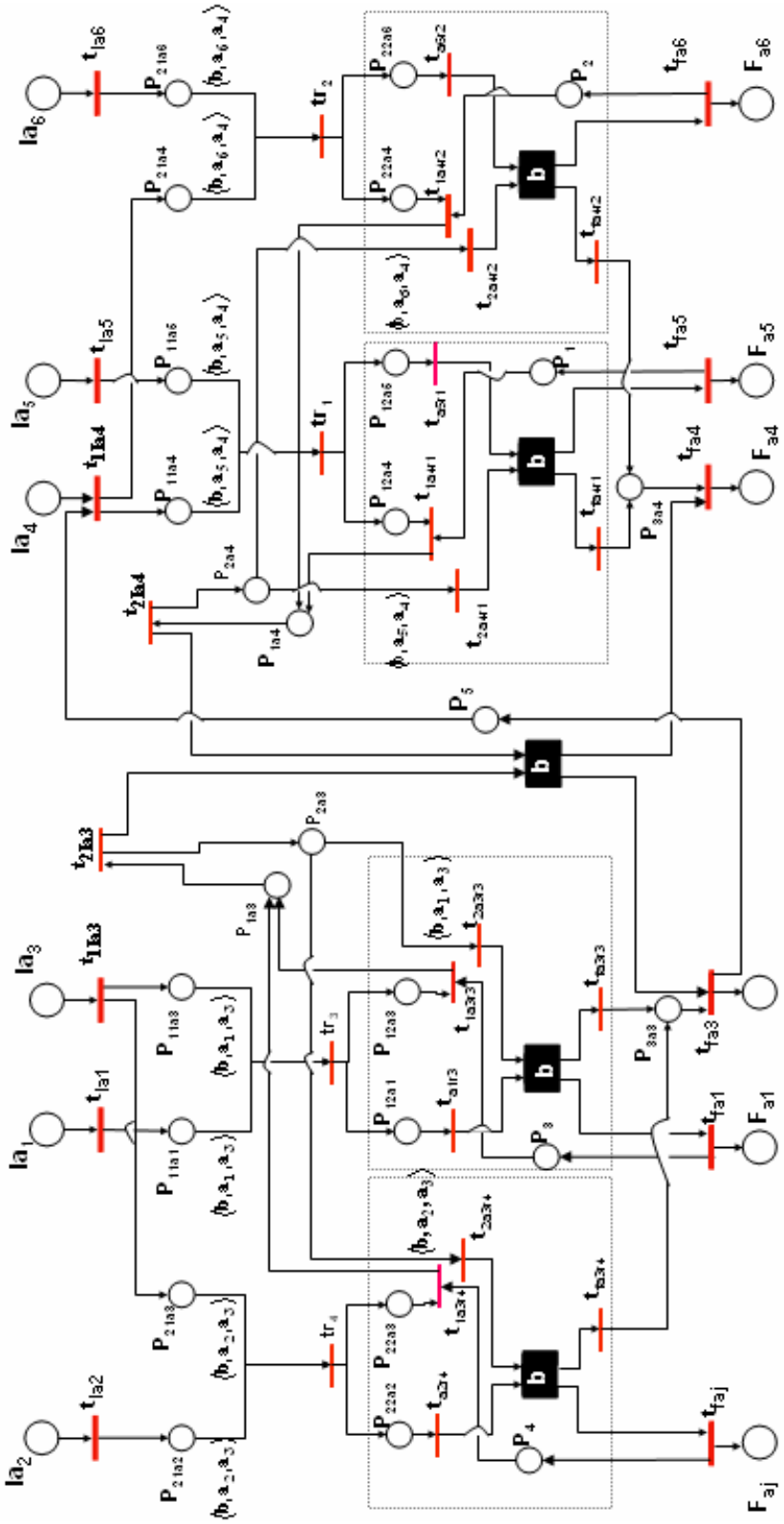


Figura 4-8 : Mecanismo de coordenação de $(\mathbf{E}, \mathbf{G})$.

4.2. Estudo de Caso 2 – Ambiente Multimídia

A metodologia **GR** usa o formalismo oferecido pelas redes de Petri Coloridas para construir o mecanismo de coordenação de uma expressão. O emprego desta extensão das redes de Petri é explorado neste trabalho em situações que exigem a modelagem de seleção de relações e/ou atividades.

A ausência de operações de seleção na especificação de uma expressão determina que seu mecanismo de coordenação é representado através de uma Rede de Petri padrão. O objetivo desta seção é apresentar um exemplo que demonstra esta última situação.

O exemplo que será apresentado ilustra uma situação envolvendo uma aplicação multimídia. O ambiente inicia apresentando um vídeo que é reproduzido simultaneamente com o áudio. Após a apresentação do vídeo1 é iniciada a reprodução do vídeo2 que acontece em simultâneo com a reprodução do áudio2. Após iniciar a exibição do vídeo2 estabelece-se uma comunicação remota por texto, vídeo e áudio com um monitor para fazer possíveis questionamentos a respeito dos assuntos tratados.

Para o problema do Ambiente Multimídia acima definiram-se oito atividades que são listadas a seguir.

Atividade	descrição
a1	Reprodução do vídeo1
a2	Reprodução do vídeo2
a3	Reprodução do áudio1
a4	Comunicação via texto
a5	Estabelecimento de comunicação com o monitor
a6	Reprodução do áudio2
a7	Entrada de vídeo usado na comunicação com o monitor
a8	Entrada e saída de áudio usado na comunicação com o monitor

Os relacionamentos entre as atividades são especificados na expressão $\mathbf{E(A,R,F)}$ dada a seguir.

$$\mathbf{A} = \{\mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3, \mathbf{a}_4, \mathbf{a}_5, \mathbf{a}_6, \mathbf{a}_7, \mathbf{a}_8\},$$

$$\mathbf{R} = \{(\mathbf{a}_3, \mathbf{a}_1), (\mathbf{a}_1, \mathbf{a}_2), (\mathbf{a}_2, \mathbf{a}_6), (\mathbf{a}_5, \mathbf{a}_2), (\mathbf{a}_5, \mathbf{a}_4), (\mathbf{a}_5, \mathbf{a}_7), (\mathbf{a}_8, \mathbf{a}_7)\},$$

F a função rotuladora de arestas dada por:

$$\mathbf{Y}_{3,1} = \{\mathbf{e}\}, \mathbf{Y}_{1,2} = \{\mathbf{b}\}, \mathbf{Y}_{2,6} = \{\mathbf{s}\}, \mathbf{Y}_{5,2}\{\mathbf{f}\},$$

$\mathbf{Y}_{5,4} = \{\mathbf{s}\}$, $\mathbf{Y}_{5,7} = \{\mathbf{s}\}$ e $\mathbf{Y}_{8,7} = \{\mathbf{s}\}$.

A Figura 4 - 9 ilustra uma representação gráfica da expressão $\mathbf{E}(\mathbf{A}, \mathbf{R}, \mathbf{F})$.

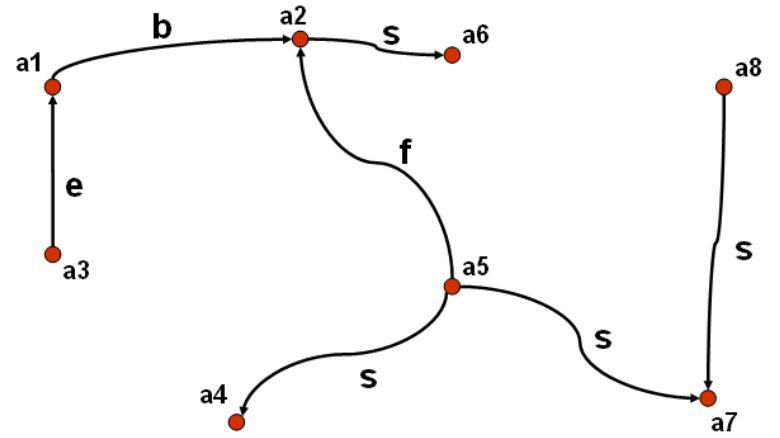


Figura 4 - 9: Representação gráfica da expressão $\mathbf{E}(\mathbf{A}, \mathbf{R}, \mathbf{F})$.

O processo de construção (Algoritmo 1) do **MC** da expressão **E** inicia-se determinando suas curvas de nível (Definição 14): $\mathbf{I}_0 = \{\mathbf{a}_3, \mathbf{a}_4, \mathbf{a}_6, \mathbf{a}_8\}$, $\mathbf{I}_1 = \{\mathbf{a}_1, \mathbf{a}_7\}$ e $\mathbf{I}_2 = \{\mathbf{a}_2, \mathbf{a}_5\}$. O próximo passo consiste em identificar e modelar as restrições temporais das estrelas de $\mathbf{I}_1$, isto é, obter os mecanismos $\mathbf{MCa}_1$ e $\mathbf{MCa}_7$.

Do Procedimento 2 tem-se que:

- $\mathbf{MCa}_1 = \mathbf{MC}_1 \oplus \mathbf{MC}_2$ e
- $\mathbf{MCa}_7 = \mathbf{MC}_3 \oplus \mathbf{MC}_4$, sendo
- $\mathbf{MC}_1$ e $\mathbf{MC}_2$ os mecanismos que estruturam as restrições temporais derivados das relações $\mathbf{e}(\mathbf{a}_3, \mathbf{a}_1)$ e $\mathbf{b}(\mathbf{a}_1, \mathbf{a}_2)$, respectivamente;
- $\mathbf{MC}_3$ e $\mathbf{MC}_4$ os mecanismos que estruturam as restrições temporais derivados das relações $\mathbf{s}(\mathbf{a}_8, \mathbf{a}_7)$ e $\mathbf{s}(\mathbf{a}_5, \mathbf{a}_7)$, respectivamente.

A representação das restrições temporais para uma relação é sempre a mesma, independentemente das atividades envolvidas. Assim, as Figura 4 - 10 (a), **(b)**, **(c)** e **(d)** apresentam os mecanismos $\mathbf{MC}_1$, $\mathbf{MC}_2$, $\mathbf{MC}_3$ e $\mathbf{MC}_4$ nesta ordem, obtidos dos diagramas apresentados na Seção 3.3.3.2. (Figura 3 - 21(a), Figura 3 - 19(a) e Figura 3 - 22 (a)).

Os mecanismos $\mathbf{MCa}_1 = \mathbf{MC}_1 \oplus \mathbf{MC}_2$ e $\mathbf{MCa}_7 = \mathbf{MC}_3 \oplus \mathbf{MC}_4$ (Figura 4 - 10 (e) e (f)) são obtidos aplicando-se o item (a) do Procedimento 1.

Como as operações de conexão das estrelas de $\mathbf{I}_1$ com as de $\mathbf{I}_0$ já foram efetuadas com o processo de construção das estrelas de $\mathbf{I}_1$; o próximo passo é gerar os mecanismos de coordenação das estrelas de $\mathbf{I}_2 = \{\mathbf{a}_2, \mathbf{a}_5\}$.

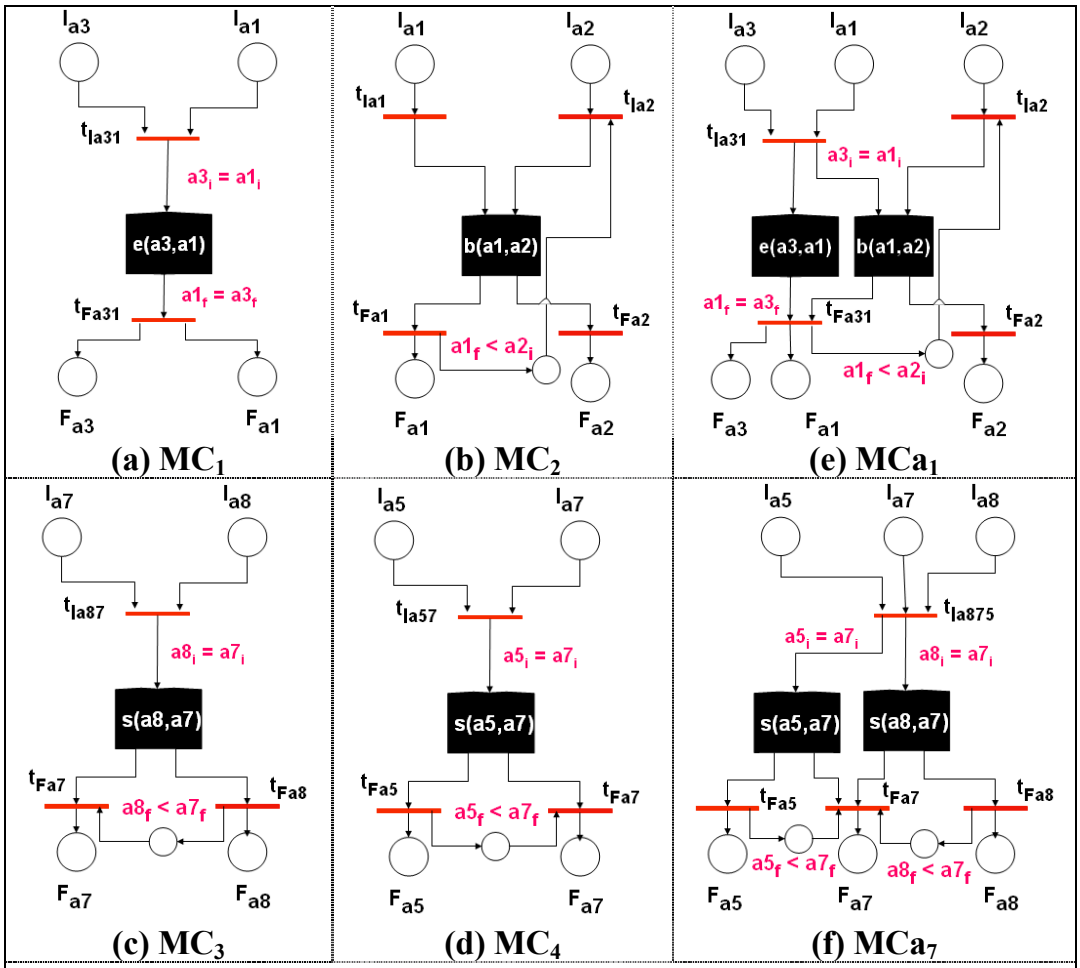


Figura 4-10: **MCs** das relações e das estrelas $\mathbf{a}_1$ e $\mathbf{a}_7$.

Novamente, do Procedimento 2 tem-se que:

- $\mathbf{MC}_2 = \mathbf{MC}_5 \oplus \mathbf{MC}_6$ e
- $\mathbf{MCa}_5 = \mathbf{MC}_7$, sendo
- $\mathbf{MC}_5$ e $\mathbf{MC}_6$ os mecanismos das relações $\mathbf{s}(\mathbf{a}_2, \mathbf{a}_6)$ e $\mathbf{f}(\mathbf{a}_5, \mathbf{a}_2)$ e
- $\mathbf{MC}_7$ o mecanismo da relação $\mathbf{s}(\mathbf{a}_5, \mathbf{a}_4)$.

Estes mecanismos estão ilustrados na Figura 4-11 (a), (b) e (c). Do Procedimento 1 item (a) obtém-se o mecanismo $\mathbf{MCa}_2$ representado graficamente na Figura 4-11 (d).

A seguir concatenam-se os mecanismos de coordenação, $\mathbf{MCa}_2$ e $\mathbf{MCa}_5$, das estrelas de $\mathbf{I}_2$ com os mecanismos de coordenação, $\mathbf{MCa}_1$ e $\mathbf{MCa}_7$, das estrelas de $\mathbf{I}_1$.

Como a estrela $\mathbf{a}_2$ é adjacente à $\mathbf{a}_1$ e a estrela $\mathbf{a}_5$ é adjacente à $\mathbf{a}_7$ (Definição 17) as operações de conexão $\mathbf{MCa}_1 \oplus \mathbf{MCa}_2$ e $\mathbf{MCa}_5 \oplus \mathbf{MCa}_7$ podem ser efetuadas. Da Proposição 4 tem-se que tanto $\mathbf{MCa}_1 \oplus \mathbf{MCa}_2$ quanto $\mathbf{MCa}_5 \oplus \mathbf{MCa}_7$ são efetuados pelo item (a) do Procedimento 1.

Assim definimos $\mathbf{MC}_8 = \mathbf{MCa}_1 \oplus \mathbf{MCa}_2$, que é obtido efetuando-se uma fusão da transição $\mathbf{t}_{Ia2}$ de $\mathbf{MCa}_1$ com a transição $\mathbf{t}_{Ia26}$ de $\mathbf{MCa}_2$ e uma fusão da transição $\mathbf{t}_{Fa2}$ de $\mathbf{MCa}_1$ com a transição $\mathbf{t}_{Fa25}$ de $\mathbf{MCa}_2$, conforme Figura 4-12 (c).

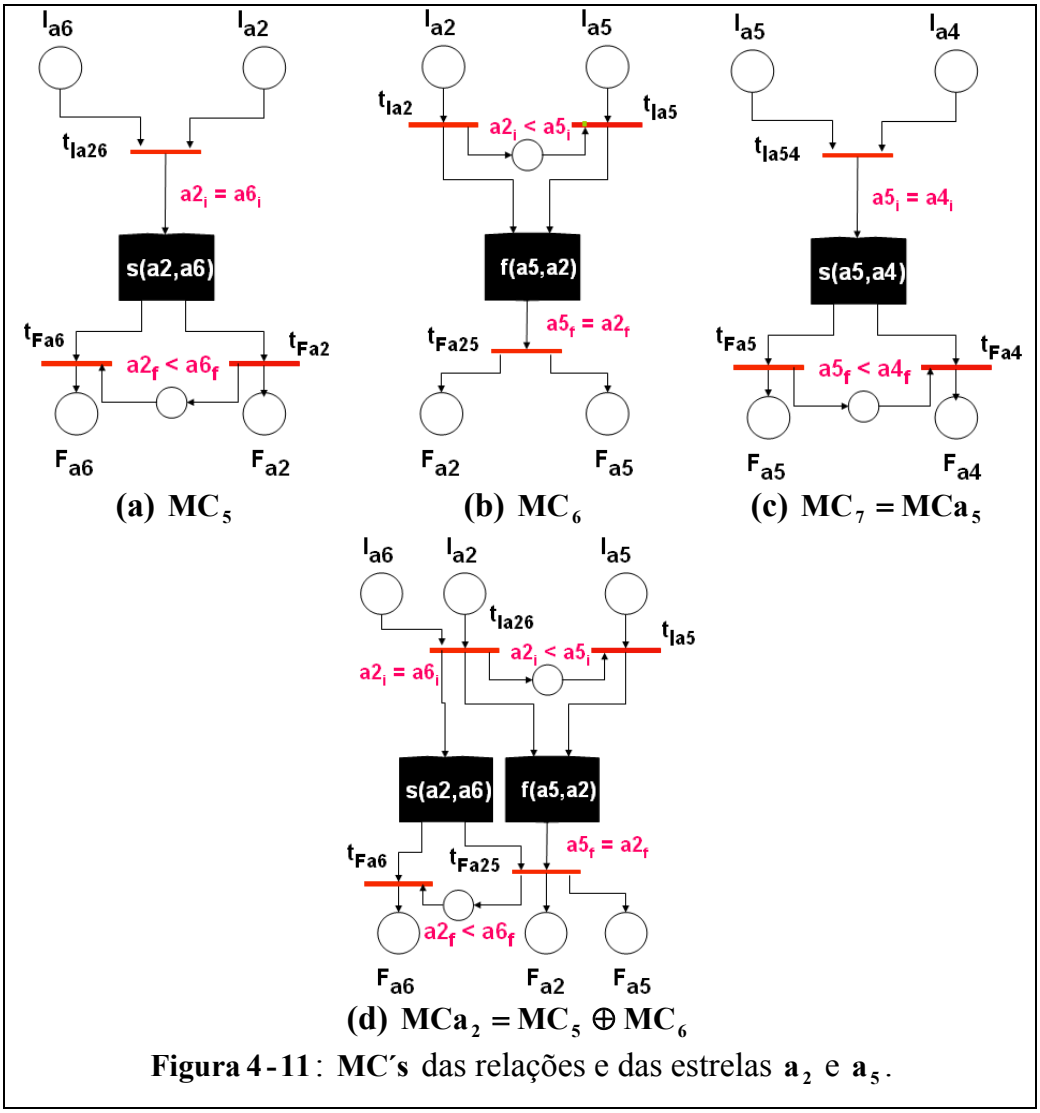


Figura 4-11: MC´s das relações e das estrelas $\mathbf{a}_2$ e $\mathbf{a}_5$.

Da mesma forma definimos $\mathbf{MC_9 = MCa_5 \oplus MCa_7}$, que é obtido efetuando-se uma fusão da transição $\mathbf{t_{Ia54}}$ de $\mathbf{MCa_5}$ com a transição $\mathbf{t_{Ia875}}$ de $\mathbf{MCa_7}$ e uma fusão da transição $\mathbf{t_{Fa5}}$ de $\mathbf{MCa_5}$ com a transição $\mathbf{t_{Fa5}}$ de $\mathbf{MCa_7}$ (ver Figura 4 - 13 (c)).

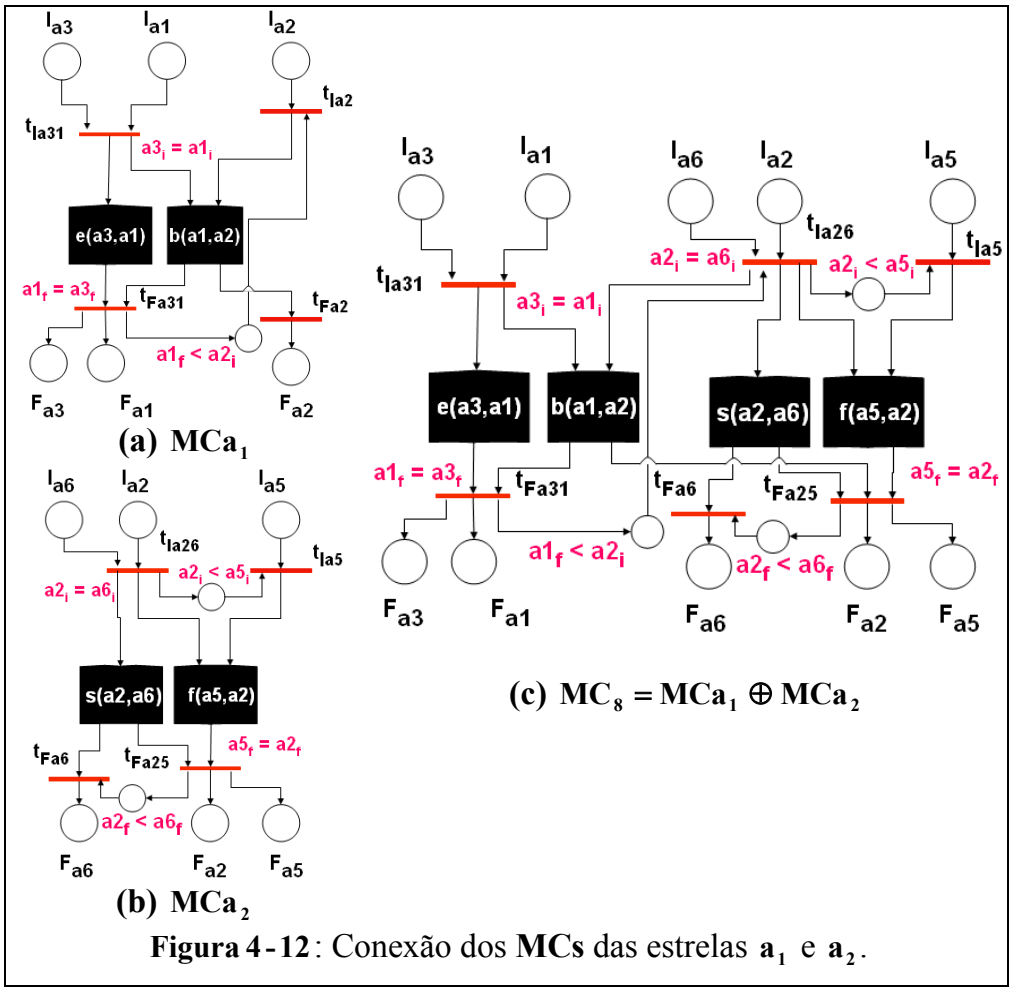


Figura 4 - 12: Conexão dos **MCs** das estrelas $\mathbf{a_1}$ e $\mathbf{a_2}$.

Finalmente o mecanismo de coordenação global **MC** da expressão $\mathbf{E(A,R,F)}$ é obtido conectando-se os **MCs** das estrelas de $\mathbf{I_2}$ que correspondem ao centro da expressão (Seção 3.3.2.). Esta operação equivale a conectar $\mathbf{MC_8}$ a $\mathbf{MC_9}$ aplicando-se o item **(a)** do **Procedimento 1**.

O $\mathbf{MC = MC_8 \oplus MC_9}$ é obtido efetuando-se uma fusão da transição $\mathbf{t_{Ia5}}$ de $\mathbf{MC_8}$ com a transição $\mathbf{t_{Ia8754}}$ de $\mathbf{MC_9}$ e uma fusão da transição $\mathbf{t_{Fa25}}$ de $\mathbf{MC_8}$ com a transição $\mathbf{t_{Fa5}}$ de $\mathbf{MC_9}$ conforme ilustra a Figura 4 - 14 .

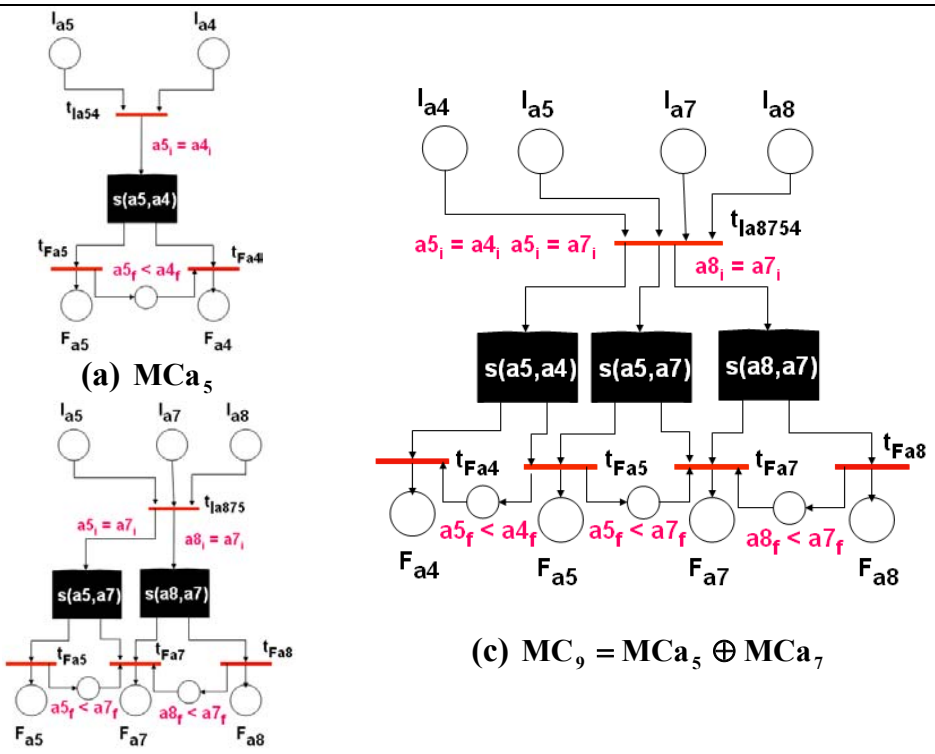


Figura 4-13 : Conexão dos **MCs** das estrelas $\mathbf{a_5}$ e $\mathbf{a_7}$.

Por fim, verifica-se que todos os mecanismos gerados na construção do **MC** e por conseguinte ele próprio, são redes de Petri padrão.

Com este exemplo procurou-se mostrar que a metodologia **GR** pode gerar mecanismos de coordenação modelados por redes de Petri padrão. Estes mecanismos são gerados a partir de expressões que não usam estruturas de seleção nem de atividade nem de relação. Portanto, tais mecanismos são aplicados a situações onde todas as atividades devem ser executadas pelo menos uma vez, isto é, todos os comportamentos especificados devem ser reproduzidos a fim de serem alcançados os objetivos pretendidos com a aplicação. Um exemplo no campo da animação por computador seria coordenar os movimentos de um grupo de dançarinos [Magalhães 98]. Considerando a coreografia a ser executada expressa por uma expressão **GR**, as atividades determinam os movimentos que cada dançarino deve realizar e as relações entre elas o momento da execução. Assim, para que a coreografia seja cumprida a contento não pode haver improvisos, ou seja, todos os passos devem ser executados. Ainda no campo da animação por computador, um outro exemplo análogo ao anterior, seria coordenar as atividades de uma orquestra. Estas atividades determinam o que cada músico deve executar e as relações da expressão o momento da execução. Para que a música seja interpretada

completamente e necessário que todas as atividades sejam executadas. Um exemplo aplicado ao trânsito de veículos é apresentado na seção a seguir.

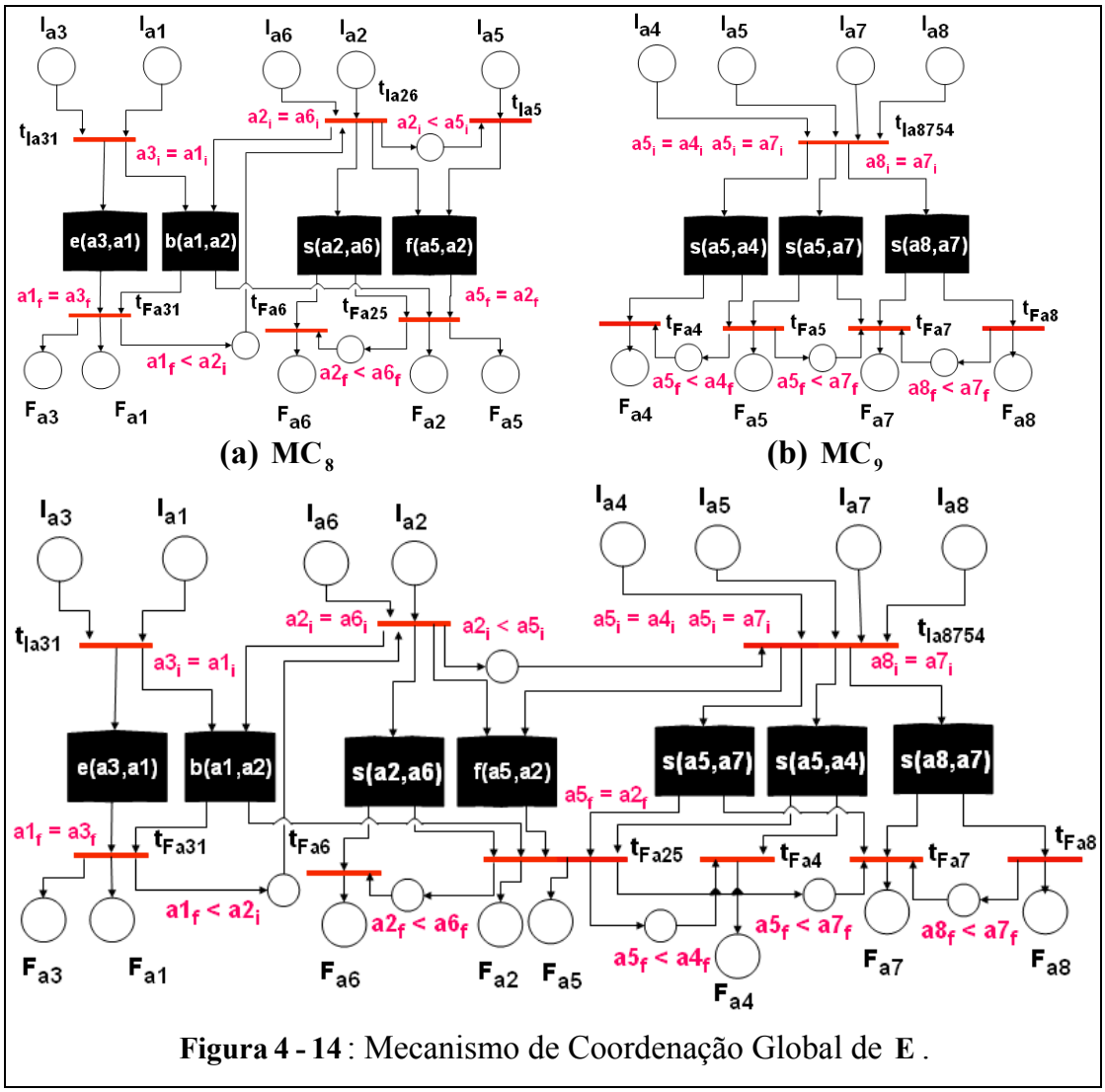


Figura 4 - 14 : Mecanismo de Coordenação Global de **E** .

4.3. Estudo de Caso 3 - Coordenando o tráfego em um sinaleiro

Neste exemplo busca-se mostrar a flexibilidade da metodologia **GR** na definição dos intervalos de tempo atribuídos às atividades. Adicionalmente, mostra-se que o mapeamento entre eventos do ambiente e as transições com reserva de fichas permitem estabelecer maior interação com o ambiente e, através deste, com o usuário da aplicação. Por exemplo, pode-se permitir que um usuário determine o fim ou início de atividades conforme o tempo desta.

Neste estudo de caso aplica-se a metodologia **GR** na solução de um problema de tráfego. Supondo um ambiente virtual, deseja-se coordenar o fluxo de carros e pessoas em uma interseção de duas avenidas de forma a evitar a colisão entre carros e entre carros e pessoas.

O mapa representado na Figura 4 - 15 ilustra o ambiente a ser coordenado. Os sinaleiros $\mathbf{S_1}$, $\mathbf{S_2}$, $\mathbf{S_3}$ e $\mathbf{S_4}$ trabalham sincronizados. Quando $\mathbf{S_1}$, e $\mathbf{S_3}$ estão "abertos", isto é, com luz verde, $\mathbf{S_2}$ e $\mathbf{S_4}$ estão "fechados", isto é, com luz vermelha ou amarela.

O estado aberto ou fechado dos sinaleiros determina o fluxo dos carros e das pessoas. Por exemplo, se $\mathbf{S_1}$ e $\mathbf{S_3}$ estão fechados então os pedestres podem atravessar do lado **A** para o lado **D** e vice-versa, os carros podem seguir na direção **AD** ou efetuarem a conversão **A**, entre outras atividades.

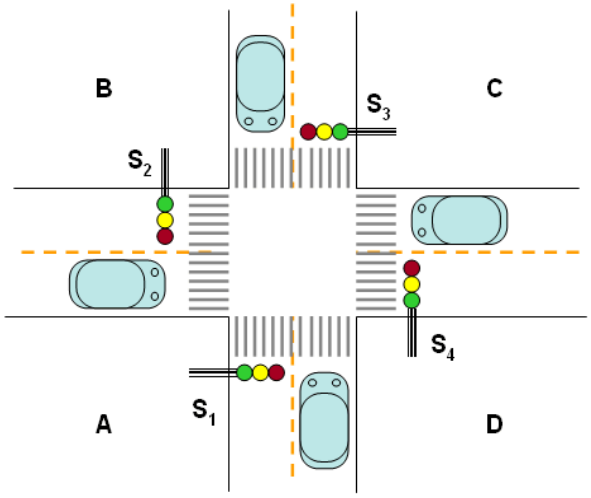


Figura 4 - 15 : Interseção de duas avenidas.

Apresenta-se a seguir o conjunto de atividades a ser coordenado.

Atividade	descrição
a1	Fecha sinaleiros **S2** e **S4**
a2	Pedestres atravessam do lado **A** para o lado **B**
a3	Pedestres atravessam do lado **B** para o lado **A**
a4	Carros seguem a direção **BA**
a5	Carros efetuam a conversão **B**
a6	Pedestres atravessam do lado **C** para o lado **D**
a7	Pedestres atravessam do lado **D** para o lado **C**
a8	Carros seguem a direção **DC**
a9	Carros efetuam a conversão **D**
a10	Abre sinaleiros **S2** e **S4**
a11	Fecha sinaleiros **S1** e **S3**

a12	Pedestres atravessam do lado **A** para o lado **D**
a13	Pedestres atravessam do lado **D** para o lado **A**
a14	Carros seguem a direção **AD**
a15	Carros efetuam a conversão **A**
a16	Pedestres atravessam do lado **B** para o lado **C**
a17	Pedestres atravessam do lado **C** para o lado **B**
a18	Carros seguem a direção **CB**
a19	Carros efetuam a conversão **C**
a20	Abre sinaleiros **S1** e **S3**

Os relacionamentos entre as atividades são especificados na expressão $\mathbf{E(A,R,F)}$ especificada a seguir e representada graficamente na Figura 4-16.

$$\mathbf{A} = \left\{ \begin{matrix} \mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3, \mathbf{a}_4, \mathbf{a}_5, \mathbf{a}_6, \mathbf{a}_7, \mathbf{a}_8, \mathbf{a}_9, \mathbf{a}_{10}, \mathbf{a}_{11}, \\ \mathbf{a}_{12}, \mathbf{a}_{13}, \mathbf{a}_{14}, \mathbf{a}_{15}, \mathbf{a}_{16}, \mathbf{a}_{17}, \mathbf{a}_{18}, \mathbf{a}_{19}, \mathbf{a}_{20} \end{matrix} \right\},$$

$$\mathbf{R} = \left\{ \begin{matrix} (\mathbf{a}_1, \mathbf{a}_2), (\mathbf{a}_1, \mathbf{a}_3), (\mathbf{a}_1, \mathbf{a}_4), (\mathbf{a}_1, \mathbf{a}_5), (\mathbf{a}_1, \mathbf{a}_6), (\mathbf{a}_1, \mathbf{a}_7), (\mathbf{a}_1, \mathbf{a}_8), \\ (\mathbf{a}_1, \mathbf{a}_9), (\mathbf{a}_1, \mathbf{a}_{10}), (\mathbf{a}_1, \mathbf{a}_{20}), (\mathbf{a}_{20}, \mathbf{a}_{11}), (\mathbf{a}_{11}, \mathbf{a}_{12}), (\mathbf{a}_{11}, \mathbf{a}_{13}), \\ (\mathbf{a}_{11}, \mathbf{a}_{14}), (\mathbf{a}_{11}, \mathbf{a}_{15}), (\mathbf{a}_{11}, \mathbf{a}_{16}), (\mathbf{a}_{11}, \mathbf{a}_{17}), (\mathbf{a}_{11}, \mathbf{a}_{18}), (\mathbf{a}_{11}, \mathbf{a}_{19}), \end{matrix} \right\},$$

F a função rotuladora de arestas dada por:

$$\mathbf{Y}_{1,2} = \{\mathbf{e}\}, \quad \mathbf{Y}_{1,3} = \{\mathbf{e}\}, \quad \mathbf{Y}_{1,4} = \{\mathbf{e}\}, \quad \mathbf{Y}_{1,5} = \{\mathbf{e}\}, \quad \mathbf{Y}_{1,6} = \{\mathbf{e}\},$$
$$\mathbf{Y}_{1,7} = \{\mathbf{e}\}, \quad \mathbf{Y}_{1,8} = \{\mathbf{e}\}, \quad \mathbf{Y}_{1,9} = \{\mathbf{e}\}, \quad \mathbf{Y}_{1,10} = \{\mathbf{m}\}, \quad \mathbf{Y}_{1,20} = \{\mathbf{e}\},$$
$$\mathbf{Y}_{20,11} = \{\mathbf{m}\}, \quad \mathbf{Y}_{11,12} = \{\mathbf{e}\}, \quad \mathbf{Y}_{11,13} = \{\mathbf{e}\}, \quad \mathbf{Y}_{11,14} = \{\mathbf{e}\}, \quad \mathbf{Y}_{11,15} = \{\mathbf{e}\},$$
$$\mathbf{Y}_{11,16} = \{\mathbf{e}\} \quad \mathbf{Y}_{11,17} = \{\mathbf{e}\} \quad \mathbf{Y}_{11,18} = \{\mathbf{e}\} \quad \text{e} \quad \mathbf{Y}_{11,19} = \{\mathbf{e}\}$$

Conforme determina o Algoritmo 1 a construção do **MC** da expressão **E** deve iniciar determinando suas curvas de nível (Definição 14) a saber:

$$\mathbf{I}_0 = \{\mathbf{a}_2, \mathbf{a}_3, \mathbf{a}_4, \mathbf{a}_5\mathbf{a}_6, \mathbf{a}_7, \mathbf{a}_8, \mathbf{a}_9, \mathbf{a}_{10}, \mathbf{a}_{12}, \mathbf{a}_{13}, \mathbf{a}_{14}\mathbf{a}_{15}, \mathbf{a}_{16}, \mathbf{a}_{17}, \mathbf{a}_{18}, \mathbf{a}_{19}\},$$
$$\mathbf{I}_1 = \{\mathbf{a}_1, \mathbf{a}_{11}\} \text{ e } \mathbf{I}_2 = \{\mathbf{a}_{20}\}.$$

Na seqüência deve-se selecionar o elemento $\mathbf{I}_1$ da partição de **A**. Para este conjunto de atividades proceder a identificação e modelagem das restrições temporais das estrelas $\mathbf{a}_1$ e $\mathbf{a}_{11}$ de forma a obter os mecanismos $\mathbf{MCa}_1$ e $\mathbf{MCa}_{11}$.

Acompanhando o Procedimento 2 obtém-se as relações que devem fazer parte do **MC** de cada estrela, cujo resultado é apresentado a seguir.

$$\mathbf{MCa}_1 = (((((((\mathbf{MC}_1 \oplus \mathbf{MC}_2) \oplus \mathbf{MC}_3) \oplus \mathbf{MC}_4) \oplus \mathbf{MC}_5) \oplus \mathbf{MC}_6) \oplus \mathbf{MC}_7) \oplus \mathbf{MC}_8) \oplus \mathbf{MC}_9) \oplus \mathbf{MC}_{10};$$

$$\mathbf{MCa_{11}} = (((((((\mathbf{MC_{10}} \oplus \mathbf{MC_{11}}) \oplus \mathbf{MC_{12}}) \oplus \mathbf{MC_{13}}) \oplus \mathbf{MC_{14}}) \oplus \mathbf{MC_{15}}) \oplus \mathbf{MC_{16}}) \oplus \mathbf{MC_{17}}) \oplus \\ \oplus \mathbf{MC_{18}}) \oplus \mathbf{MC_{19}}$$

sendo $\mathbf{MC_1}, \cdots, \mathbf{MC_{10}}$ os mecanismos de coordenação das relações $\mathbf{e(a_1, a_2)}$, $\mathbf{e(a_1, a_3)}$, $\mathbf{e(a_1, a_4)}$, $\mathbf{e(a_1, a_5)}$, $\mathbf{e(a_1, a_6)}$, $\mathbf{e(a_1, a_7)}$, $\mathbf{e(a_1, a_8)}$, $\mathbf{e(a_1, a_9)}$, $\mathbf{e(a_1, a_{20})}$ e $\mathbf{m(a_1, a_{10})}$, respectivamente. E $\mathbf{MC_{11}}, \cdots, \mathbf{MC_{19}}$ os mecanismos de coordenação das relações $\mathbf{e(a_{11}, a_{12})}$, $\mathbf{e(a_{11}, a_{13})}$, $\mathbf{e(a_{11}, a_{14})}$, $\mathbf{e(a_{11}, a_{15})}$, $\mathbf{e(a_{11}, a_{16})}$, $\mathbf{e(a_{11}, a_{17})}$, $\mathbf{e(a_{11}, a_{18})}$ $\mathbf{e(a_{11}, a_{19})}$ e $\mathbf{m(a_{20}, a_{11})}$, respectivamente.

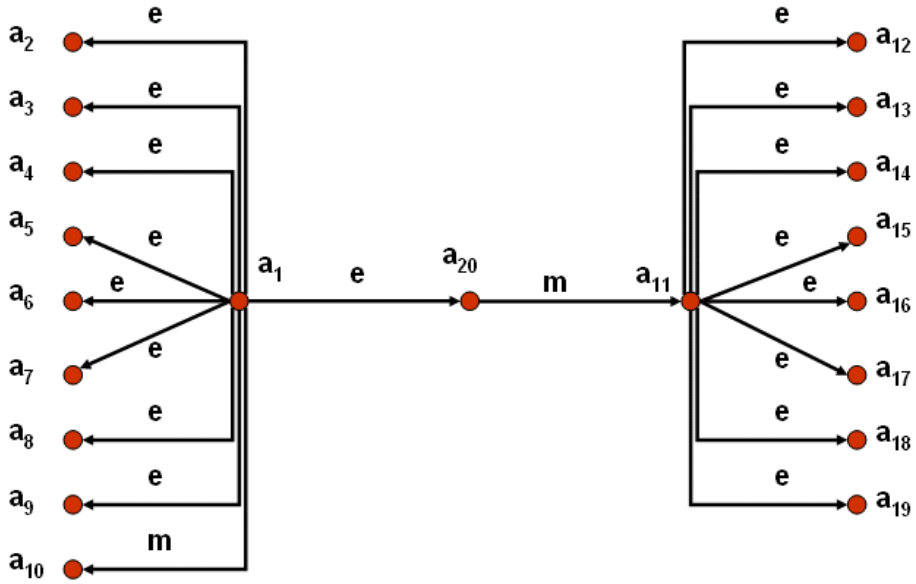


Figura 4-16 : Representação gráfica da expressão **E**.

Os mecanismos $\mathbf{MC_1}, \cdots, \mathbf{MC_9}$ são construídos a partir do diagrama apresentado na Figura 3-22 (a) e o mecanismo $\mathbf{MC_{10}}$ a partir do diagrama da Figura 3-22 (b). As Figura 4-17 (a) e (b) mostram os mecanismos obtidos para $\mathbf{MC_1}$ e $\mathbf{MC_2}$ e a Figura 4-17 (c) ilustra o mecanismo $\mathbf{MC_1} \oplus \mathbf{MC_2}$ resultante da conexão de $\mathbf{MC_1}$ com $\mathbf{MC_2}$ (Procedimento 1 item a).

O mecanismo $\mathbf{MC_1} \oplus \mathbf{MC_2}$ deve ser conectado ao mecanismo $\mathbf{MC_3}$ (Procedimento 1, item a) e este aos outros mecanismos das relações da estrela $\mathbf{a_1}$, isto é:

$$\mathbf{MCa_1} = (((((((\mathbf{MC_1} \oplus \mathbf{MC_2}) \oplus \mathbf{MC_3}) \oplus \mathbf{MC_4}) \oplus \mathbf{MC_5}) \oplus \mathbf{MC_6}) \oplus \mathbf{MC_7}) \oplus \mathbf{MC_8}) \oplus \mathbf{MC_9}) \oplus \mathbf{MC_{10}}$$

obtendo-se ao final o mecanismo ilustrado na Figura 4-18.

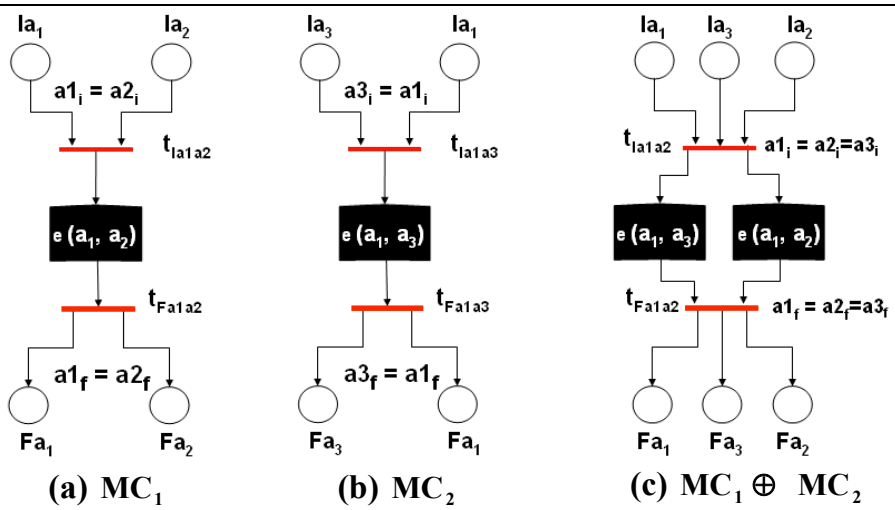


Figura 4-17 : Modelagem e conexão de duas das relações da estrela $\mathbf{a_1}$.

O mecanismo da estrela $\mathbf{a_{11}}$, isto é, $\mathbf{MCa_{11}}$ também é construído a partir dos esquemas apresentados na Figura 3-20 (a) e (b) e da operação de conexão determinada pelo Procedimento 1 item (a). O mecanismo resultante $\mathbf{MCa_{11}}$ é ilustrado na Figura 4-19.

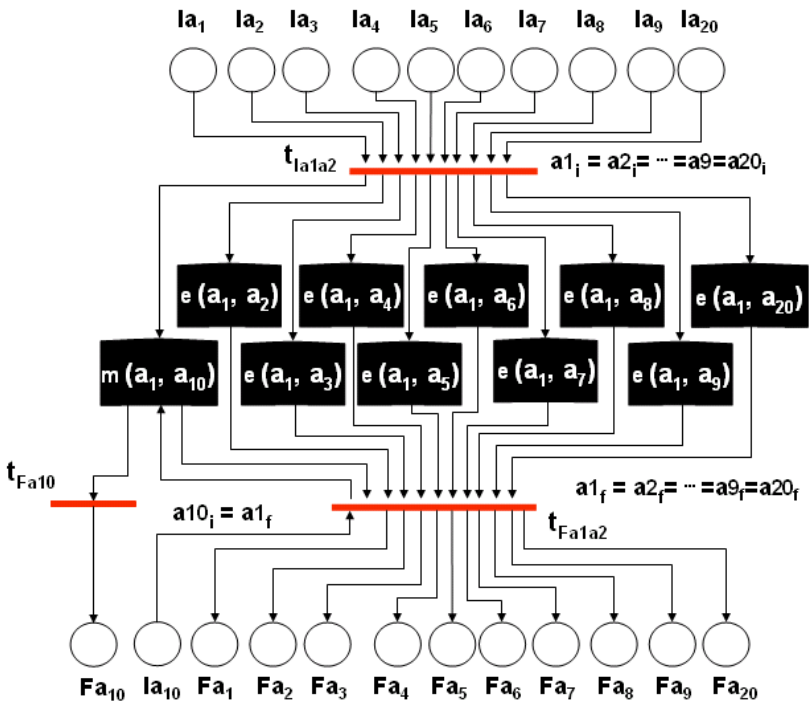


Figura 4-18: **MC** da estrela $\mathbf{a_1}$- $\mathbf{MCa_1}$.

Como os mecanismos de coordenação das estrelas de $\mathbf{I}_1$ também modelam as relações envolvendo as atividades da curva de nível $\mathbf{I}_0$ não é necessário efetuar a operação de conexão estipulada no passo 3 do Algoritmo 1, passando-se então a modelagem da estrela $\mathbf{a}_{20}$ da curva de nível $\mathbf{I}_2$.

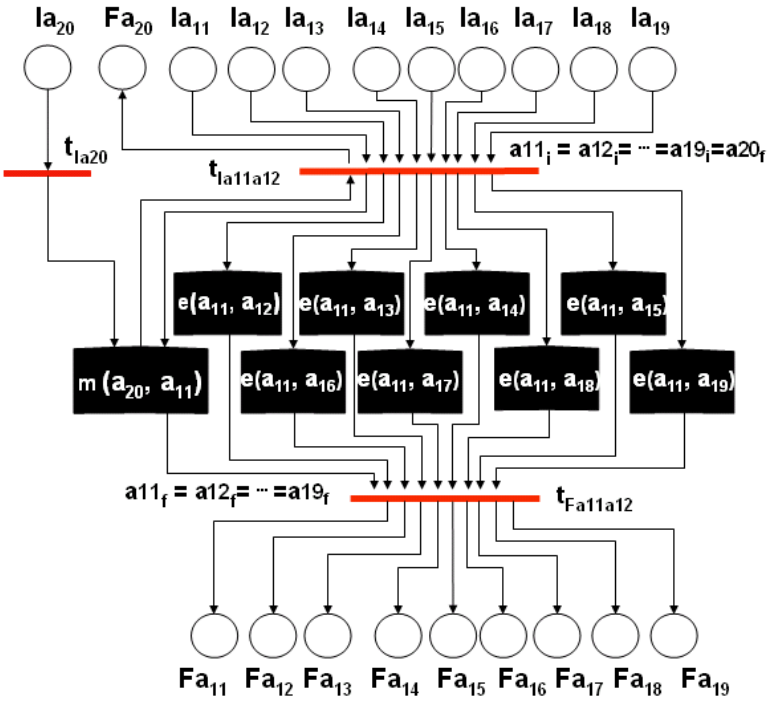


Figura 4-19: **MC** da estrela $\mathbf{a}_{11}$- $\mathbf{MCa}_{11}$.

Seguindo o Procedimento 2 conclui-se que o mecanismo de coordenação da estrela $\mathbf{a}_{20}$, isto é $\mathbf{MCa}_{20}$ corresponde unicamente a modelagem da atividade $\mathbf{a}_{20}$ (Figura 4-20).

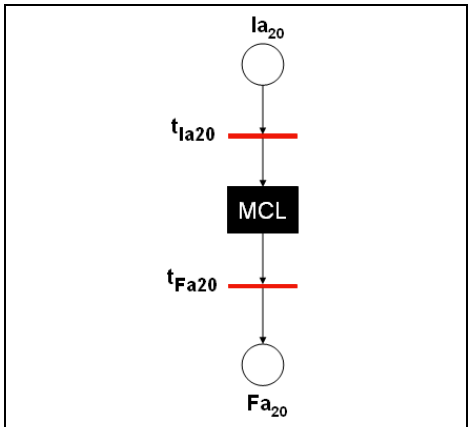


Figura 4-20: **MC** da estrela $\mathbf{a}_{20}$.

O próximo passo desta segunda e última iteração é a conexão dos **MCs** das estrelas adjacentes das curvas de nível $\mathbf{I}_1$ e $\mathbf{I}_2$ resultando no **MC** da expressão **E**, analiticamente:

$$\mathbf{MC} = \left(\mathbf{MCa}_{20} \oplus \mathbf{MCa}_1\right) \oplus \mathbf{MCa}_{11}.$$

No entanto, a operação $\mathbf{MCa}_{20} \oplus \mathbf{MCa}_1$ é desnecessária, pois a atividade $\mathbf{a}_{20}$ já foi modelada no $\mathbf{MCa}_1$, logo

$$\mathbf{MCa}_{20} \oplus \mathbf{MCa}_1 = \mathbf{MCa}_1.$$

Por fim, para obter-se o **MC** da expressão, Figura 4 - 21, é suficiente efetuar a operação $\mathbf{MCa}_1 \oplus \mathbf{MCa}_{11}$ aplicando-se o item (a) do Procedimento 1.

Neste exemplo, o tempo gasto para a execução de cada atividade pode ser determinado tendo em vista duas situações: intervalo de tempo com duração determinada ou não determinada. Por exemplo, pode-se definir que os sinaleiros ficarão abertos durante 30 unidades de tempo ou que os sinaleiros ficarão fechados até que os pedestres terminem de atravessar a faixa de segurança.

Para a primeira situação, intervalo de tempo com duração determinada, devem-se então atribuir o valor 30 às transições com reserva de fichas (encapsuladas nas "caixas pretas" do **MC** da Figura 4 - 21) associadas as atividades que determinam abertura e fechamento dos sinaleiros, ou seja as atividades **a1**, **a10**, **a11** e **a20**.

Para a segunda situação, intervalo de tempo com duração não determinada, o tempo em que os sinaleiros ficarão fechados ou abertos não depende do (evento) tempo despendido gerenciado pelo mecanismo de coordenação, mas de um evento determinado pelo ambiente informando que os pedestres terminaram de atravessar a faixa de segurança. Estes eventos devem ser associados às atividades de interesse, isto é, as transições com reserva de fichas correspondentes a estas atividades, no caso as atividades **a2**, **a3**, **a6** e **a7** ou **a12**, **a13**, **a16** e **a17**. As ocorrências destes eventos determinam que as transições com reserva de fichas devem enviar suas fichas para os lugares de saída, indicando o fim do tempo de execução destas atividades.

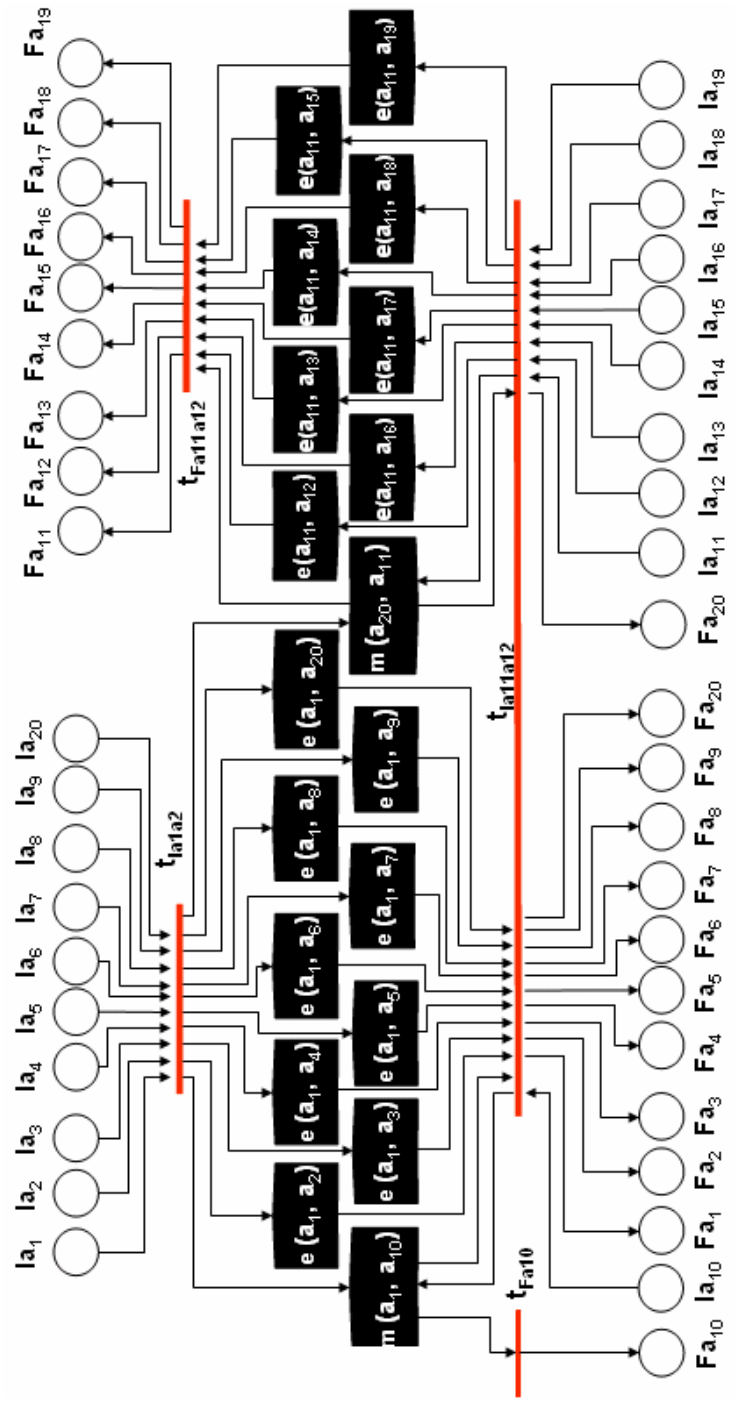


Figura 4 - 21: **MC** da expressão $\mathbf{E(A, R, F)}$ - Caso 3

4.4. Considerações do Capítulo

Um dos pontos de destaque da metodologia **GR** é a sua capacidade de gerar automaticamente a partir de uma expressão seu mecanismo de coordenação livre de inconsistências temporais. Entre os benefícios resultantes desta capacidade tem-se:

- a não interferência do projetista na construção da rede de Petri que modela o mecanismo de coordenação, evitando a inserção de possíveis erros;
- o uso da metodologia não está vinculado a um conhecimento prévio dos conceitos e formalismo pertinentes às redes de Petri;
- que uma alteração no nível da especificação do problema, isto é, na expressão, não é origem de mais trabalho para o projetista no nível da construção do mecanismo de coordenação, isto é, o projetista se preocupa em definir o relacionamento entre as atividades e não com os detalhes de construção do mecanismo de coordenação. O que pode traduzir-se em fator de motivação para experimentar melhorias nesta especificação;
- que não é necessário explorar processos de análise e verificação de inconsistências temporais, pois as regras para especificação dos comportamentos temporais garante esta consistência.

Tanto o nível de abstração **N1**, nível da especificação, quanto o nível de abstração **N2**, nível da coordenação, Figura 3 - 3, oferecem uma representação analítica e uma representação gráfica. A simplicidade da representação gráfica de uma expressão facilita a visualização global dos relacionamentos entre as atividades e a identificação de eventuais mudanças de relacionamentos entre elas.

Como comentado nos casos apresentados neste capítulo os mecanismos de coordenação possibilitam:

- selecionar comportamentos alternativos, isto é, a estrutura de seleção permite escolher uma atividade dentre um conjunto de atividades e para cada uma destas pode-se aplicar novamente a estrutura de seleção. Ainda, para cada duas atividades pode-se definir uma estrutura de seleção para escolher uma relação dentre um conjunto de relações;
- associar eventos do ambiente às transições com reserva de fichas de modo a determinar o início ou fim das atividades. Esta capacidade pode ser usada para estabelecer maior interação entre o usuário e a aplicação.

Os mecanismos de coordenação também podem ser usados na fase de prototipação do ambiente computacional. A simulação e análise destes protótipos podem auxiliar no refinamento da especificação inicial efetuando-se as alterações desejáveis antes da fase de implementação. Questões relativas a análise dos

protótipos obtidos estão fora do escopo deste trabalho, pois o mesmo concentrou-se na geração de mecanismos de coordenação livres de inconsistências temporais. Outras questões, como a coordenação de recursos e a existência de ciclos na especificação dos comportamentos temporais, são discutidas no capítulo de conclusões a seguir.

5. CONCLUSÕES

Este trabalho apresentou uma metodologia para descrever e coordenar interdependências em um conjunto de atividades realizadas em um ambiente computacional. Foi apresentado igualmente um algoritmo para automatizar a geração de Mecanismo de Coordenação (MC), livre de inconsistências temporais, em tempo linear no número de atividades.

Mecanismos de coordenação local gerenciam dependências temporais isoladamente e a decisão de início ou fim da execução de uma atividade é tomada em função das atividades envolvidas diretamente na dependência temporal, ou seja, local. Neste tipo de política fica a cargo do projetista da aplicação identificar e modelar as condições globais que devem ser aplicadas a uma atividade em particular. Dessa forma, qualquer alteração na especificação dos comportamentos temporais requer do projetista uma nova análise das condições globais a fim de evitar-se a inserção de alguma inconsistência. Por este fato é desejável ter-se metodologias que automatizem o processo de modelagem dos **MC** a partir da descrição dos comportamentos.

Entre as contribuições da metodologia apresentada está o tratamento global dado ao processo de coordenação, isto é, uma atividade só é executada se nenhuma relação de interdependência é violada.

A metodologia **GR** permite automatizar o processo de geração dos **MC** – mecanismos de coordenação através de ferramentas de modelagem e do algoritmo para identificação e modelagem das condições globais. Além de diminuir o trabalho do projetista, esta automação elimina os erros de análise na determinação de tais condições e padroniza o processo de modelagem das condições globais.

A política de coordenação adotada por **GR** explora tanto as vantagens de uma coordenação global quanto local. No nível local, o **MCL** explora o conceito de modularização permitindo mudar os mecanismos locais sem interferir no **MC** global. Esta política de coordenação potencializa o uso da metodologia apresentada neste texto em ambientes colaborativos, pois gerar mecanismos de coordenação que atuem ao mesmo tempo no nível global e no nível local é uma das dificuldades encontradas na coordenação de atividades colaborativas [Cruz 02].

A inserção de um nível de coordenação permitiu fazer a distinção entre a coordenação do trabalho e como este trabalho é executado. A metodologia **GR** não determina como o trabalho é feito, mas o que deve ser feito e quando. Esta característica é pertinente a uma metodologia para construção de mecanismos de coordenação visto que as atividades são dependentes da aplicação enquanto os

mecanismos de coordenação excedem esta fronteira. Em outras palavras, mudar como um fragmento do trabalho (atividade) é feito não deve implicar obrigatoriamente em mudanças no nível da coordenação. Outra vantagem da abordagem por níveis de abstração é desvincular o uso da metodologia **GR** de um conhecimento prévio do formalismo usado para modelar os **MC** (no caso redes de Petri).

O uso de conceitos da teoria dos grafos no processo de descrição e modelagem dos comportamentos temporais propiciou a principal contribuição deste trabalho, o desenvolvimento de um algoritmo linear no número de atividades envolvidas em um ambiente computacional.

O conceito de restrição temporal e a operação de conexão $\oplus$ favoreceram um modelo analítico para os mecanismos de coordenação permitindo que estes fossem descritos por expressões composta por mecanismos de coordenação parciais.

O emprego das redes de Petri coloridas na modelagem dos mecanismos de coordenação favoreceu o encapsulamento do **MCL** e das representações das condições de sincronização de início e fim das atividades. A escolha por esta extensão das redes de Petri permitiu gerar redes mais compactas na modelagem das operações de seleção entre relações e seleção entre atividades. Se a expressão a ser modelada não apresenta opção de seleção então a rede gerada é uma rede de Petri padrão.

Na seção seguinte discutem-se a modelagem de comportamentos não temporais e as expectativas para trabalhos futuros.

5.1 Discussões, avaliações e trabalhos futuros

A execução de uma atividade também pode estar condicionada à disponibilidade de recursos. Estes recursos podem ser produzidos por outras atividades ou recursos que estão fora do escopo da aplicação. Por exemplo, uma atividade pode produzir dados que são compartilhados por outras atividades. Uma atividade pode necessitar de dispositivos para realizar seu trabalho, como uma impressora, serviços de rede, entre outros.

Para o caso em que os recursos são produzidos no escopo da aplicação, as dependências também podem ser coordenadas pela metodologia **GR** descrevendo-se as relações entre as atividades de acordo com as necessidades de recursos além das restrições temporais.

Esta dependência de recurso pode ser vista como uma relação do tipo produtor-consumidor, onde uma ou mais atividades produzem e uma ou mais atividades consomem. Esta relação pode ser traduzida para uma ou mais das dependências temporais apresentadas.

As primitivas **meets** ou **before** podem ser usadas para representar dependências de recursos na forma seqüencial, conforme ilustra a Figura 5 - 1.

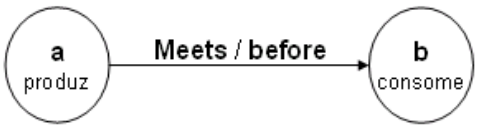


Figura 5 - 1: Relação produtor-consumidor **1x1**

Outra possibilidade é o compartilhamento de recursos onde uma atividade produz para outras consumirem. Na Figura 5 - 2 a atividade **a** produz e as atividades **b** e **c** são servidas do produto. Esta dependência pode ser descrita usando a primitiva **meets**.

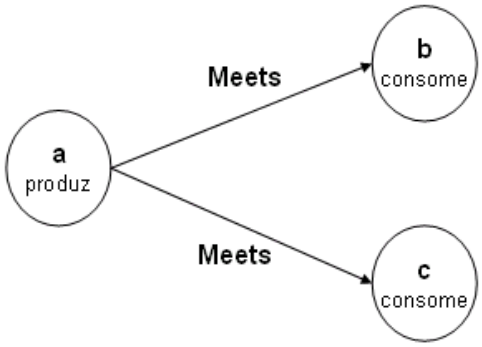


Figura 5 - 2: Relação produtor-consumidor **1xn**.

Por fim, como exemplo de representação da combinação de recursos, onde várias atividades produzem para uma atividade, pode-se usar a primitiva **meets**. A Figura 5 - 3 ilustra o caso **2x1** para as atividades **a**, **b** e **c**.

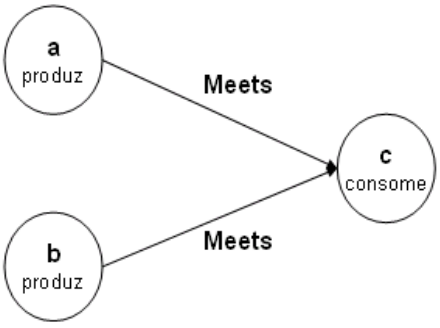


Figura 5 - 3: relação produtor-consumidor **nx1**.

Embora a metodologia **GR** não trate diretamente as dependências de recursos que estão fora do escopo da aplicação, existem duas possibilidades para esta questão.

A primeira é tratá-la também na especificação isto é, atividades que competem por recursos limitados não devem ser executadas em paralelo, o que reduz a capacidade de representação da metodologia. Uma forma para tratar esta questão é expandir o modelo para atividade, por exemplo, como o proposto por [van der Aalst 96] e inserir mecanismos de coordenação para gerenciar as dependências de recursos como por exemplo, os propostos por [Raposo 00a]. Os trabalhos [Raposo 01a] e [Raposo 01b] abordam, usando uma política de coordenação local, a questão da coordenação de atividades e recursos em ambientes virtuais colaborativos empregando-se tecnologia de componentes de software.

Outro assunto a abordar refere-se a adaptar o algoritmo de identificação e modelagem também para expressões cíclicas consistentes e investigar propriedades que permitam transformar expressões inconsistentes em consistentes. No Capítulo 3 mostrou-se que uma expressão acíclica é consistente e que esta é uma condição suficiente, no entanto expressões cíclicas podem ser consistentes, por exemplo, um ciclo onde todas as dependências são definidas pela primitiva **equals**.

Uma abordagem possível é explorar as relações de transitividade derivadas das primitivas temporais apresentadas pela Definição 7. A expectativa é determinar técnicas para identificação de informações redundantes na especificação dos comportamentos temporais. Se existem informações redundantes, derivadas das primitivas que formam um ciclo, então é possível suprimir o ciclo eliminando tais informações.

Pretende-se explorar em trabalhos futuros a metodologia **GR** na coordenação de comportamentos não temporais como, por exemplo, aqueles definidos por dependências espaciais. Uma possibilidade é adotar para estes outros conjuntos de dependências a mesma estratégia de modelagem, isto é, pré-definir mecanismos de coordenação para cada conjunto de dependências e usar uma operação de conexão para obter mecanismos de coordenação mais complexos.

Finalmente, deseja-se investigar as possíveis redundâncias inseridas nos mecanismos de coordenação em função do seu processo de automatização e estabelecer regras para tornar estes mecanismos mais compactos.

REFERÊNCIAS

[Agostine 97] A. Agostini and G. De Michelis**.** Simple models for articulating complex work processes. *Cooperation Technologies Laboratory, DSI - University of Milano,* Italy.1997. http://ccs.mit.edu/klein/cscw98/.

[Allen 83] J. F. Allen. Maintaining Knowledge about Temporal Interval. *Communications of the ACM*, 23(11): 832-843. November 1983.

[Allen 84] J. F. Allen**.** Towards a General Theory of Action and Time, *Artificial Intelligence*, 23 (1984) 123-154.

[Alonso 97] G. Alonso, D. Agrawal, A. El Abbadi, and C. Mohan. Functionality and Limitations of Current Workflow Management Systems. *IEEE Expert, 12(5)*, September-October 1997.

[Baggetun 00] R. Baggetun and A. Morch. Coordination as Resourse in Collaborative Telelearning. *Proceedings of IRIS 23. Laboratorium for Interaction Technology, University of Trollhättan Uddevalla.* August, 2000. http://iris23.htu.se/proceedings/PDF/55final.PDF

[Bannon 91] L**.** J. Bannon and K. Shimidt. CSCW: four Characters in Search of a context. In J. M. Bowers and S. D. Benford (Eds.). *Studies in Computer Supported Cooperative Work*, pp. 3-16. North-Holland, 1991.

[Bannon 94] L**.** J**.** Bannon. CSCW Challenging Perspectives on Work and Technology. *Information Technology & Organizational Change*. Nijenrode Business School. The Netherlands. 28-29 April, 1994.

[Benford 01] S**.** Benford, C. Greenhalgh, T. Rodden and J. Pycock. Collaborative Virtual Environments. *Communications of The ACM*. 44(7), pp 79-85. July, 2001.

[Bull 92] S. A. Bull. *Flow Path Functional Specification*. September , 1992.

[Carstensen 96] P**.** H**.** Carstensen. Computer Supported Coordination. *Ph. D, Risø*. National Laboratory, Roskilde, pp. 250 April 1996.

[Cassandras 93] C. G. Cassandras. *Discrete Event Systems: Modeling and Performance Analysis*. Aksen Associates. 1993.

[Chen 02] W**.** Chen and K. S. Decker. Coordination Mechanisms for Dependency Relationships among Multiple Agents. *AAMAS'02*, July 15-19, Bologna, Italy, 2002. www.eecis.udel.edu/decaf/papers/**AAMAS2002AlternMech**.ps.

[Ciancarini 99] P. Ciancarini. Coordination Technologies for Internet Agents. *Nordic Journal of Computing*, 6, pp. 215-240, 1999.

[Courtiat 95] J.P. Courtiat., M. Diaz, R. C. Oliveira and P. Senac. Formal Models for the Description of Timed Behaviors of Multimedia and Hypermedia Distributed Systems

[Crowston 94] K. Crouston. A Taxonomy Of Organizational Dependencies and Coordination Mechanisms. *Center for Coordination Science at the Massachusetts Institute of Technology*. August 1994. http://ccs.mit.edu/papers/CCSWP174.html.

[Crowston 00] K. Crowston. The evolution of resource allocation mechanisms for space allocation in transportation systems. *Center for Science and Technology Syracuse at Syracuse University*. November, 2000. http:// citeseer.nj.nec.com.

[Cruz 02] A. J. A Cruz, A. B. Raposo and L. P. Magalhães. Coordination in Collaborative Environments - A Global Approach. *7th International Conference on Computer Supported Cooperative Work in Design - CSCWD* 2002, p.25 - 30. Rio de Janeiro, Brazil. 2002.

[Dellarocas 94] C. N. Dellarocas, J. Lee, T. W., Malone, K. Crowston and B. Pentland. Using a Process Handbook to Design Organizational Processes. *In Proceedings, AAAI Spring Symposium on Computational Organization Design*, pp. 50-56. Stanford, CA. March 21-23, 1994.

[Dellarocas 96] C. N. Dellarocas. A Coordination Perspective on Software Architecture: Towards a Design Handbook for Integrating Software Components. PhD Thesis, Dept. of Electrical Engineering and Computer Science – MIT, 1996.

[Duda 95] A. Duda and C. Keramane. Structured temporal composition of multimedia data**.** *In Proc. IEEE International Workshop on Multimedia-Database-Management Systems***,** Blue Mountain Lake**,** August 1995. http://citeseer.nj.nec.com/duda95structured.html.

[Duda 97] A. Duda and C. Keramane. Operator Based Composition of Structured Multimedia Presentations. *COST 237 Workshop*,1-17. 1997. http://citeseer.nj.nec.com/451825.html.

[Ellis 99] C. Ellis and J. Wainer. Groupware and Computer Supported Cooperative Work. *Chapter in Multiagent systems : a modern approach to distributed artificial intelligence*, MIT Press, Cambridge, Mass., 1999.

[Ferraro 97] A. M. Ferraro and E. H**.** Rogers Petri Nets in the Evaluation of Collaborative Systems. *The 1997 IEEE International Conference on Systems, Man, and CyberneticsHyatt*. Orlando, Florida, USA. October 12-15, 1997.

[Ferreira 99] A. B. H. Ferreira. *Dicionário Aurélio Eletrônico – século XXI*. Versão 3.0. Lexikon Informática Ltda. Novembro 1999.

[Franklin 96] S. Franklin and A. Graesser. *"Is it an Agent, or just a Program?: A Taxonomy for Autonomous Agents"*. Proceedings of the Third International Workshop on Agent Theories, Architectures and Languages, Springer-Verlag, 1996.

[Georgakopoulos 95] D. Georgakopoulos and M. F. Hornick and A. P. Sheth**.** An Overview of Workflow Management: From Process Modeling to Workflow Automation Infrastructure. *Distributed and Parallel Databases*,3 (2): 119-153. 1995.

[Herskind] S. Herskind and P. A. Nielsen**.** Coordination Mechanisms in Ephemeral Organizations. *Proceedings of IRIS 20*. Hanko, Norway, 1997. http://iris.informatik.gu.se/conference/iris20/55.htm .

[Holvoet 96] T. Holvoet and P. Verbaeten**.** Synchronization specifications for agents with net-based behavior descriptions. In Proceedings of CESA '96 IMACS Conference, Symposium on Discrete Events and Manufacturing Systems, pp 613-618. Lille, France, July 1996.

[Hsu 03] P. Hsu, Y. Chang and Y. Chen. STRPN: A Petri-Net Approach for Modeling Spatial-Temporal Relations between Moving Multimedia Objects. IEEE Transactions on Software Engineering, Vol 29, No. 1, January 2003.

[Jensen 97] K. Jensen. A Brief Introduction to Coloured Petri Nets. In: E. Brinksma (ed.): Tools and Algorithms for the Construction and Analysis of Systems. Proceeding of the TACAS'97 Workshop, Enschede, The Netherlands 1997, Lecture Notes in Computer Science Vol. 1217, Springer-Verlag 1997, 203-208.

[Ljungberg 95] J. Ljungberg and P. Holm. *Speech Acts On Trial* . In proceedings of Third Decennial Conference: Computers in Context Joining Forces in Design. Aarhus, Denmark, 1995

[Magalhães 98] L. P. Magalhães, A. B. Raposo and I. L. M. Ricarte. Animation Modeling with Petri Nets. *Computer & Graphics*, 22(6): 735-743. 1998. Pergamon.

[Malone 93] T. W. Malone, J. Lee and B. Pentland. Tools for inventing organizations: Toward a handbook of organizational processes. *In Proceedings of the 2nd IEEE Workshop on Enabling Technologies Infrastructure for Collaborative Enterprises***.** Morgantown, WV, April 20-22. 1993.

[Malone 94] T. W. Malone T. W. and K. Crowston. The Interdisciplinary Study of Coordination. *ACM Computing Surveys*, *26*(1): 87-119**.** 1994 (March).

[Marazakis 97] M. Marazakis, C. Nikolaou, and D. Papadakis. "Workflow Management Systems: State-of-the-Art", 1997. Working Paper - available via URL http://citeseer.nj.nec.com/marazakis97workflow.html

[Mates 72] B. Mates. Elementary Logic. *New York Oxford University Press*. Second Edition, 1972.

[Maurer 93] H. Maurer. Um Panorama dos Sistemas de Hipermídia e Multimídia. *Mundos Virtuais e Multimídia*. Editora LTC. 1993.

[Michelis 94] G. De Michelis and M. A. Grasso. Situating Conversations Within the Language/Action Perspective: The Milan Conversation Model. *Computer Supported Cooperative Work*, pp 89-100. 1994

[Millar 99] R. J. Millar, J. R. P. Hanna and S. M. Kealy. A review of behavioral animation. *Computer & Graphics*, 23, pp. 127-143, 1999.

[Moeckel 00] A. Moeckel. Modelagem de Processos de Desenvolvimento em Ambiente de Engenharia Simultânea: Implementações com as Tecnologias workflow e BSCW. *Dissertação de Mestrado*. CEFET-PR. Novembro, 2000.

[Moldt 97] D. Moldt and F. Wienberg. Multi-Agent-Systems base on Coloured Petri Nets. *In Proceedings of the 18th International Conference on Application and Theory of Petri Nets*. Toulouse, June 23-27, France, 1997.

[Murata 89] T. Murata. Petri Nets: properties, analysis and applications, Proc. Of the IEEE, 77(4) (1989) 542-580.

[Orlikowski 92] W. J. Orlikowski. LEARNING FROM NOTES: Organizational Issues in Groupware Implementation. *MIT Sloan School Working Paper #3428-92*, *Center for Coordination Science Techincal Report #134*. May 1992.

[Papadopoulos 00] G. A. Papadopoulos. Models and Technologies for the Coordination of Internet Agents: A Survey. *Chapter in Coordination of Internet Agents, Springer Verlag*, 2000.

[Ramamoorthy 80] C. V. Ramamoorthy and G. S. Ho. Performance evaluation of asynchronous concurrent systems using Petri Nets. *IEEE transactions in Software Engineering*, 6(5): 440-449. September 1980.

[Raposo 00a] A. B. Raposo. Coordenação em Ambientes Colaborativos Usando redes de Petri, Tese de Doutorado DCA/FEEC/UNICAMP, outubro, 2000.

[Raposo 00b] A. B. Raposo, L. P. Magalhães and I. L. M. Ricarte. Petri Nets Based Coordination Mechanisms for Multi-Workflow Environments. *International Journal of Computer Systems Science & Engineering*. Special Issue on Flexible Workflow Tecnology Driving the Networked Economy. CRL Publishing. September 2000.

[Raposo 01a] A. B. Raposo, A. J. A. da Cruz, C. M. Adriano and L. P. Magalhães. Coordination Components for Collaborative Virtual Environments. *Computers & Graphics*, vol. 25, no. 6,pp. 1025-1039. December 2001. Special Issue on Artificial Life: Towards A New Generation of Computer Animation. Pergamon Press, Holland (ISSN 0097 8493).

[Raposo 01b] A. B. Raposo, C. M. Adriano, A. J. A. da Cruz and L. P. Magalhães. A Component-Based Infrastructure for the Coordination of Collaborative Activities in Virtual Environment. *SVR 2001 - 4th SBC Symposium on Virtual Reality*, p.127 - 138. Florianópolis, Brazil. SBC, 2001.

[Russell 03] S. Russell and P. Norvig, *Solution Manual for Artificial Intelligence: A Modern Approach*. 2nd Edition, Prentice Hall, 2003.

[Schimidt 91] K Schimidt. Computer Support for Cooperative Work in Advanced Manufacturing. *International Journal of Human Factors in Manufacturing*, 1(4): 303-320. October 1991.

[Schimidt 96] K Schimidt and C Simone. Coordination Mechanisms: Towards a conceptual foundation of CSCW systems design. *Computer Supported Cooperative Work: The Journal of Collaborative Computing*, 5(2-3): 155-200. 1996.

[Senac 95] P. Senac, R. Willrich and M. Diaz. Hypermedia Synchronization Modeling: a Case Study. *ED-MEDIA'95 World Conf. on Educational Multimedia and Hypermedia*. *Graz*. 1995. www.citeseer.nj.nec.com

[Szwarcfiter 86] Szwarcfiter, J L Grafos e algoritmos computacionais, 2ª edição, Editora Campus, Rio de Janeiro 1986.

[Tanenbaum 95] A. S. Tanenbaum,. Distributed operating system, Prentice-Hall, 1995.

[van der Aalst 94] W. M. P. van der Aslst, K. M. van Hee and G. J. Houben. Modeling and analyzing workflow using a Petri-net based approach. *Pro. Of the 2nd Workshop on Computer-Supported Cooperative Work, Petri nets and related formalisms*, pp 31-50. 1994.

[van der Aalst 96] W. M. P. van der Aslst. Petri-net-based workflow management software. *In Proc. NSF Wkshp. Workflow and Process Automation in Info. Syst., 1996*.

[Vazirgiannis 99] M. Vazirgiannis and I. Kostalas. Specifying and Authoring Multimedia Scenarios. *IEEE MultiMedia*, 6(3): 24-27. 1999.

[Verbeek 04] H.M.W. Verbeek. and W.M.P. Van Der Aalst and Akhil Kumar. XRL/Woflan: Verification and Extensibility of an XML/Petri-Net-Based Language for Inter-Organizational Workflows. *Information Technology and Management*, 5(1-2), 65–110. 2004.

[Villanova 00] M. Villanova, N. Belkhatir and H. Martin. A Temporal Process Model for Web Multimedia Applications. 2000. www.citeseer.nj.nec.com

[Wahl 94] T. Wahl and K. Rothermel. Representing time in multimedia systems**.** *In Proc. IEEE International Conference on Multimedia Computing and Systems***.** Boston, MA. May 1994**.**

[Weber 97] M. Weber, G. Partsch, S. Hoeck, G. Schneider, A. Scheller-Houy, J. Schweitzer. Integrating Synchronous Multimedia Collaboration into Workflow Management**.** *In proceedings of GROUP'97*, pp. 281-290. 1997.

[Weyns 02] D. Weyns and T. Holvoet. A Colored Petri Net for Multi-Agent Application. *In Proceedings of MOCA'02*, pp.121-140, Aarhus, Denmark August, 2002.

[Winograd 86] T. Winogrand and F. Flores. Understanding Computers and Cognition: A New Foundation for Design**.** Ablex, 1986**.**

[Yoon 98] K. Yoon and P. B. Berra. Interactive Temporal Model for Interactive Multimedia Documents. In Proceedings *IW- MMDBMS*, 136-144. 1998. citeseer.nj.nec.com/311457.html.

[Zaidi 99] A. K Zaidi. On temporal Logic Programming Using Petri Nets. *IEEE Transactions on Systems, MAN, and Cybernetics – Parte A: Systems and Humans*, 29(3). May 1999.

APÊNDICE A – ARTIGOS PUBLICADOS

Apresenta-se neste apêndice uma coleção de três artigos publicados pelo autor e que estão relacionados ao contexto do trabalho documentando os estágios de evolução do mesmo.

[**Cruz 02**] A. J. A Cruz, A. B. Raposo and L. P. Magalhães. Coordination in Collaborative Environments - A Global Approach. *7th International Conference on Computer Supported Cooperative Work in Design - CSCWD* 2002, p.25 - 30. Rio de Janeiro, Brazil. 2002.

[Raposo 01a] A. B. Raposo, A. J. A. da Cruz, C. M. Adriano and L. P. Magalhães. Coordination Components for Collaborative Virtual Environments. *Computers & Graphics*, vol. 25, no. 6,pp. 1025-1039. December 2001. Special Issue on Artificial Life: Towards A New Generation of Computer Animation. Pergamon Press, Holland (ISSN 0097 8493).

[Raposo 01b] A. B. Raposo, C. M. Adriano, A. J. A. da Cruz and L. P. Magalhães. A Component-Based Infrastructure for the Coordination of Collaborative Activities in Virtual Environment. *SVR 2001 - 4th SBC Symposium on Virtual Reality*, p.127 - 138. Florianópolis, Brazil. SBC, 2001.

Coordination in Collaborative Environments – A Global Approach

Adailton J. A. da Cruz	Alberto B. Raposo	Léo P. Magalhães
[1]*DCA – FEEC*	*Tecgraf*	*DCA – FEEC*
Univ. of Campinas - Brazil	*Computer Science Dept.*	*Univ. of Campinas - Brazil*
[2]*CEUD – UFMS - Brazil*	*PUC-Rio - Brazil*	*leopini@dca.fee.unicamp.br*
ajcruz@dca.fee.unicamp.br	*abraposo@tecgraf.puc-rio.br*	

Abstract

In this work we present a methodology to express both analytically and graphically the interdependencies among tasks realized in a collaborative environment. For each interdependency expression, a coordination mechanism is built, modeling the global behavior of the environment, i.e., the structure that ensures the realization of the tasks according to the established interdependencies.

1. Introduction

An important aspect of collaborative work is the notion of tasks interdependencies [5]. These interdependencies are normally positive, in the sense that each participant wants the works of others to succeed. However, they are not always harmonious. It is necessary for coordination between tasks to exist in order to guarantee the efficiency of the collaboration. Without coordination, there is the risk that participants would get involved in conflicting or repetitive tasks. Coordination, in this context, is defined as "the act of managing interdependencies between activities performed to achieve a goal" [3].

In this sense, coordination in collaborative environments is managed by coordination mechanisms, defined as a "coordinative protocol with an accompanying artifact, such as, for instance, a standard operating procedure supported by a certain form" [7].

One of the main challenges related to the coordination of collaborative activities is to develop coordination mechanisms that are, at the same time, global and localized. The global part of the coordination mechanism should be able to collect, from the complex relationships among tasks in the collaborative environment, all the conditions that enable or not the beginning of the tasks. The localized part should be responsible to coordinate the pre-authorized execution of tasks, following the kind of relationship established for these tasks.

This paper introduces a methodology called RG (Relationships Graph) that describes, analytically and graphically, the relationships among collaborative tasks and constitutes the basis upon which global coordination mechanisms can be assembled from localized ones. The next section discusses some general issues regarding coordination mechanisms. The RG methodology is presented in Section 3 and a set of coordination mechanisms built using this methodology is presented in Section 4. Section 5 presents the conclusions of this work.

2. Temporal Coordination Mechanisms

The nature of the coordination (i.e., temporal, causal, etc.) is established by the set of relationships allowed among tasks. For example, if those relationships are temporal, then the tasks' behaviors are coordinated in relation to their execution times (e.g., task A must be executed *before* task B and *at the same time of* task C). On the other hand, if the relationships are causal, then tasks are coordinated based on events (e.g., if task A occurs, then task B and task C must also occur).

The execution of an activity occurs in a time interval and may be considered as the result of a set of tasks (atomic actions). Consequently, these tasks are related by the time parameter. The non-execution of one or more tasks may harm the execution of the whole activity, depending on the degree of interdependency of the involved tasks.

The coordination of the realization of interdependent tasks in collaborative environments requires mechanisms that should be able to model those interdependencies and ensure that they will not be violated, enabling that the collaborative activity be fully, partially, or not executed. If all the tasks that compose the activity are authorized to execute, then the activity is fully executed. The coordination mechanism authorizes a task only when its realization does not violate any defined interdependency.

In the following sections we are going to present a methodology to describe activities and to procedurally obtain the coordination mechanisms from these descriptions.

3. The RG Methodology

The RG methodology consists in obtaining a set E of expressions that describes, from a set of relations (primitives) R, the interdependencies in a group of tasks. These tasks are related to each other composing a collaborative activity. This approach does not restrict the number of relations a task may have with another.

Examples of activities adequate for this methodology are the construction of a multimedia presentation, and the assembly of a product composed of several pieces that become available during the assembly process. In the first case, tasks may be the presentation of texts, the reproduction of audios, videos, images, among others, which need to be synchronized. In the second example, tasks are the connections of the different pieces.

Once the relations among tasks are described by the set E, it is necessary to define the coordination mechanism for each expression $e_i \in E$. The elements of E have both an analytical and a graphical representation. In the following sections it will be shown how an expression e_i may be mapped onto a connex graph.

3.1. The Set of Relationships in RG

The set of relations R adopted in this work is based on the temporal logic proposed by Allen [1]. He proved that there is a set of primitive and mutually exclusive relations that could be applied over time intervals (i.e., any pair of time intervals are necessarily related by one and only one of Allen's relations). From Allen's first order predicate logic, we have chosen the 7 primitives that constitutes R (Fig. 1).

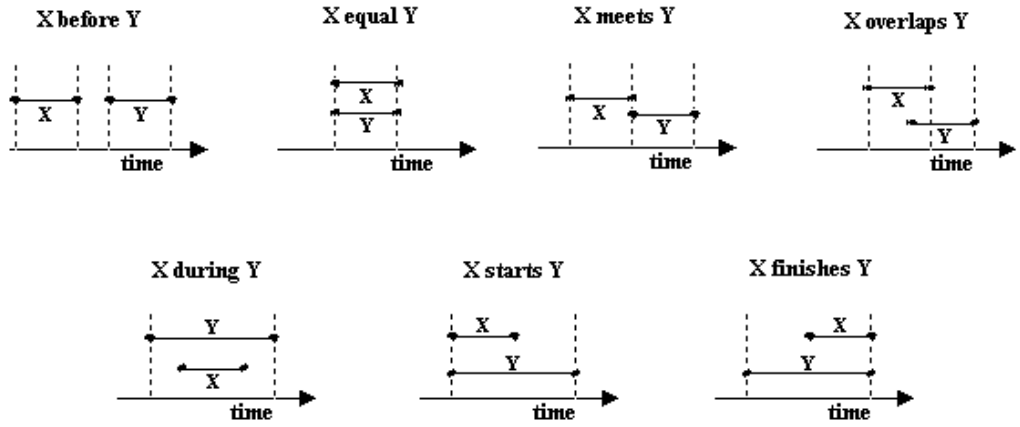


Figure 1. Set *R* of the 7 relations presented in [1].

The fact of being applied over time intervals (and not over time instants) made the relations of Fig. 1 suited for task coordination purposes, because tasks, although being the atomic actions of collaborative activities, are non-instantaneous operations.

3.2. The Expressions in RG

The primitives of *R* are binary relations, i.e., they relate only two tasks. However, it is possible to overcome this limitation, allowing that a single task be related to *k* other tasks by means of those primitives.

In order to describe an expression formed by *n* tasks t_1, t_2, …, t_n in the RG methodology we adopt the notation of the Graph Theory used by [8] as described below.

Definition 1: An expression *e(T,R)* is composed of a finite non-empty set *T* of tasks and a set *R* of labeled ordered pairs with distinct elements of *T*. The labels of the elements of *R* are defined by associating them to one of the following unitary sets, which respectively indicates the relations *before*, *during*, *equal*, *finishes*, *meets*, *overlaps* and *starts*: *{b}*, *{d}*, *{e}*, *{f}*, *{m}*, *{o}* and *{s}*.

The expression *e(T,R)* has a graphical representation where tasks $t_i \in T$, $i = 1, 2, ..., n$, correspond to distinct points of the plane located in arbitrary positions. The elements of *R* correspond to labeled arcs that connect two distinct points of the plane given by the ordered pair. Fig. 2 illustrates an example for n=12.

In an expression *e(T,R)* a task t_i has a degree k if it is related with k other tasks. For instance, in Fig. 2, $degree(t_3) = 3$.

The expression *e(T,R)* is called cyclic if there exists a sequence of *k* tasks, $1 \leq k < n$, where we start in a task t_i and return to it following that sequence in the graph. In this paper we are investigating acyclic expressions.

An expression *e(T,R)* does not generates an inconsistency if it is acyclic, i.e., the n-1 primitives do not generate an unrealizable configuration. This fact is guaranteed by the mutual exclusion and exhaustiveness properties of the primitives of *R* [9]:

Property 1: Given two tasks, *X* and *Y*, there is an $r_i \in R$ such that $X\ r_i\ Y$ or $Y\ r_i\ X$ is true (exhaustiveness).

Property 2: If $X\ r_i\ Y$, where $r_i \in R$, then there is no $r_j \neq r_i$, $r_j \in R$, such that $X\ r_j\ Y$ is true (mutual exclusion).

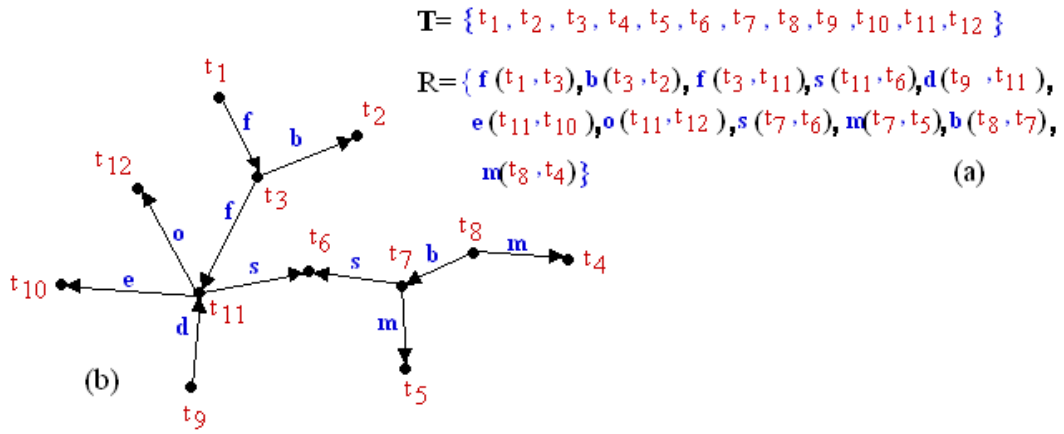


Figure 2. (a) An expression *e(T,R)*. **(b)** Its graphical representation

4. Coordination Mechanisms for Expressions

The coordination mechanism for an expression in *E* is constructed in two phases. In the first one, called Global Coordination Mechanism (GCM), the conditions that must be satisfied to authorize the beginning of the tasks are modeled. These conditions are established by the primitives that define the expression. In the second phase, called Local Control Mechanism (LCM), it is modeled the mechanism that guarantees the execution of the temporal relation established for two tasks.

Analyzing the example in Fig. 2, we can see that, for example, task t_{12} is related only to t_{11} (degree(t_{12})=1) and, therefore, does not need to satisfy any global condition. However, degree(t_{11})=5, which means that this task must satisfy some global conditions. For example, the beginning of t_{11} must be authorized simultaneously to the beginning of t_{10} (since t_{11} equal t_{10}), t_6 (t_{11} starts t_6) and t_7 (t_7 starts t_6). Another global condition for t_{11} is that it must start some time after the end of t_8 (since t_8 before t_7, and the beginning of t_7 must be simultaneous to that of t_{11}, as shown above). This is enough to give an idea that the determination of global conditions may not be easily accomplished only by looking at the graph of *e(T,R)*. The following sections will present a procedural approach to find the global conditions.

4.1. GCM – Global Coordination Mechanism

The process of a GCM construction adopts an "outside-in" approach, i.e., it starts with the most external tasks (degree=1), going to the most internal ones (higher degrees). This approach is justified because tasks with degree=1 do not need to satisfy global conditions and generally represent the majority of tasks in an expression.

Hence, given an expression e_1, we obtain a new expression e_2 by eliminating all tasks with degree=1. The derived expression e_2 also has a set of tasks with degree=1. Geometrically, we have a star of *k* legs, where the center is a task with degree=k and the legs are the arcs connecting it with its *k* related tasks. A star is defined as a sub-expression of *E* corresponding to a task *t* related to *k* other tasks (*degree(t)=k*) – see Fig. 3.

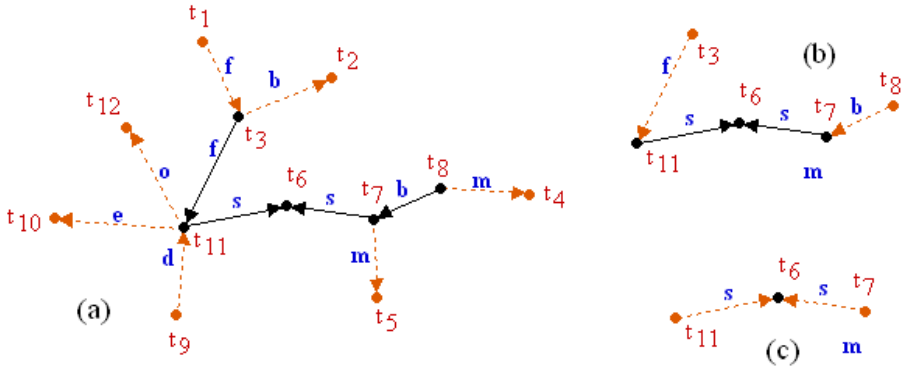


Figure 3. (a) First iteration, indicating tasks with degree=1 that will be removed from the expression. **(b)** and **(c)** following iterations.

The next step is to determine the global conditions for the tasks of degree=1 in e_2 regarding the *k* tasks related to them. In the example of Fig. 3, these tasks are t_3 and t_8.

The next iteration repeats the same steps, i.e., from e_2 we obtain an expression e_3 eliminating all tasks with degree=1 in e_2. Then we determine the global conditions for the tasks with degree=1 in e_3 regarding the *k'* tasks related to them, excepting those representing the center of the stars analyzed in the previous iteration. For example, in Fig. 3 (c), which corresponds to the expression e_3, the tasks with degree=1 are t_{11} and t_7. From the *k'* tasks related to t_{11}, task t_3 is the center of the star analyzed in the previous iteration.

In the second iteration two levels of stars appear; a more external one, generated in the first iteration and a more internal star, generated in the second iteration. In Fig. 3, the first iteration generates stars with centers t_3 and t_8, while the second iteration generates stars with centers t_{11} and t_7. Continuing with the process, we need to connect the stars of these both levels, by considering the relation with the centers of two adjacent stars.

The iterations are executed until the expression e_i, $i<n$, has one or two tasks. It can be demonstrated that this process is consistent and always finishes with one or two tasks. The algorithm for this process is described below:

Given an expression e(T,R) with n tasks;

Determine the degree of all tasks;

While there is a task t with degree(t)=1
Eliminate tasks t with degree(t)=1;
Generate stars for tasks t' with degree(t')=1;
Connect these stars with stars of the previous iterations;

4.1.1. Global Conditions.

Global conditions (GCs) are conditions imposed to a task in order to guarantee the logic of the primitives that compose the expression. For example, the beginning of t_{11} in the expression of Fig. 3 (a) must be authorized simultaneously with the beginning of t_6, t_7 and t_{10}.

The determination of GCs follows the algorithm presented in the previous section, i.e., it is established the global conditions for the stars generated in iterations *i* and *i+1*, and then the conditions corresponding to the connections of adjacent stars.

The adopted strategy for the determination of GCs is to construct a map *M* with all possible global conditions. This way, based on *M*, it is possible to determine the conditions that are pertinent to any relation in a star.

The map *M* is elaborated evaluating 14 forms of relationships possible for a task *a*. These possibilities corresponds to the 7 primitives of *R* and their respective inverse relationships. This occurs because $r(a,b) \neq r(b,a)$. For example, *a before b* is different from *b before a*. The only exception is when *r=equal*. Fig. 4 describes these forms of relationships, where the order in which the tasks are related are indicated by the arrow.

The GCs related to task *a* are more easily identified if the tasks related to it are positioned, according to their respective interdependencies, over a timeline. This process generates a consistency graph, as shown in Fig. 5.

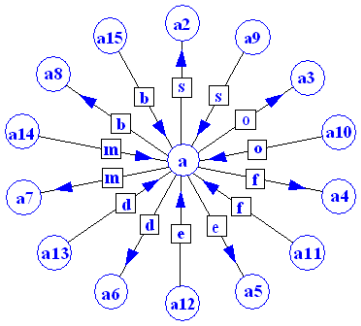


Figure 4. All possible relations involving a task.

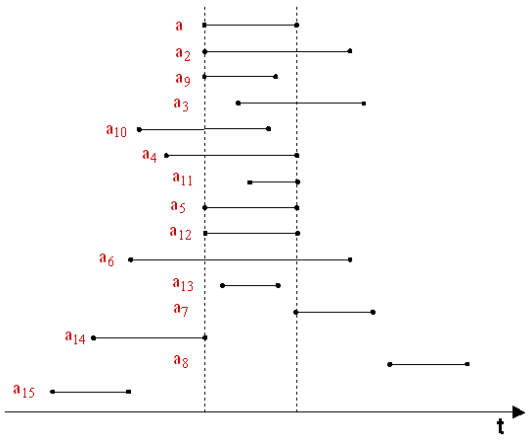


Figure 5. Consistency graph for a task.

Based on the consistency graph for task *a*, as shown in Fig. 5, the list of GCs used in the specification of the map M is described below:

c1) *a* must start simultaneously to *a2*;
c2) *a* must start simultaneously to *a9*;
c3) *a3* may start if *a* has already started;
c4) *a* may start if *a10* has already started;
c5) *a* may start if *a4* has already started;
c6) *a11* may start if *a* has already started;
c7) *a* must start simultaneously to *a5*;
c8) *a12* must start simultaneously to *a*;
c9) *a* may start if *a6* has already started;
c10) *a13* may start if *a* has already started;
c11) *a* may start if *a7* is ready to begin, but *a7* may only start at the end of *a*;
c12) *a14* may start if *a* is ready to begin, but *a* may only start at the end of *a14*;
c13) *a8* may start only after the end of *a*;
c14) *a* may start only after the end of *a15*;
c15) the end of *a* must be simultaneous to that of *a4*, *a5*, *a11*, and *a12*;
c16) the conditions to the end of the other tasks are controlled by their LCMs.

4.2. LCM – Local Coordination Mechanism

The LCM connects two related tasks, guaranteeing that a single interdependency between them will not be violated. The LCM is responsible for telling each task when it may or must start and finish its execution. It is called "local" because it coordinates only one relation between two tasks, without knowledge of GCs related to each of them. In order to implement the LCMs we use a set of Petri net-based coordination mechanisms proposed in [4].

A detailed explanation of the LCMs is out of the scope of this paper. What is important here is that LCMs may be viewed as black boxes connecting two tasks in map *M*. In the Petri net-based map, each of the possible forms of relationship of a task may be modeled according to the schema presented in Fig. 6.

In the schema of Fig. 6, task *a* is the center of the star and has a temporal relation *r(a,b)* or *r(b,a)* with task *b*. In this schema, task *b* is associated to two places, one transition and a box representing the LCM for this relation. Place *stb* indicates that *b* is ready to start and place *ftb* indicates the end of its execution. Transition *tb* receives all the conditions that *b* must satisfy and, when fired, indicates to the LCM that the global conditions for *b* are satisfied and it is authorized to begin. The LCM then assumes the coordination of the relation between *b* and the center of the star (task *a*).

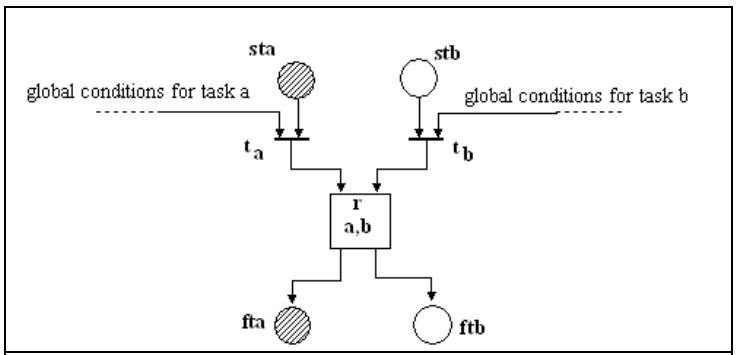


Figure 6. Schema to connect two related tasks.

4.3. The Coordination Map

The map *M* is the Petri net representation of all possible relationships involving a task *a* (as in Fig. 4). Its main goal is to show the "worst case" situation, from which other situations may derived. The map (Fig. 7) is constructed by applying the schema of Fig. 6 and the list of GCs (Section 4.1.1) to each of the 14 tasks related to task *a*.

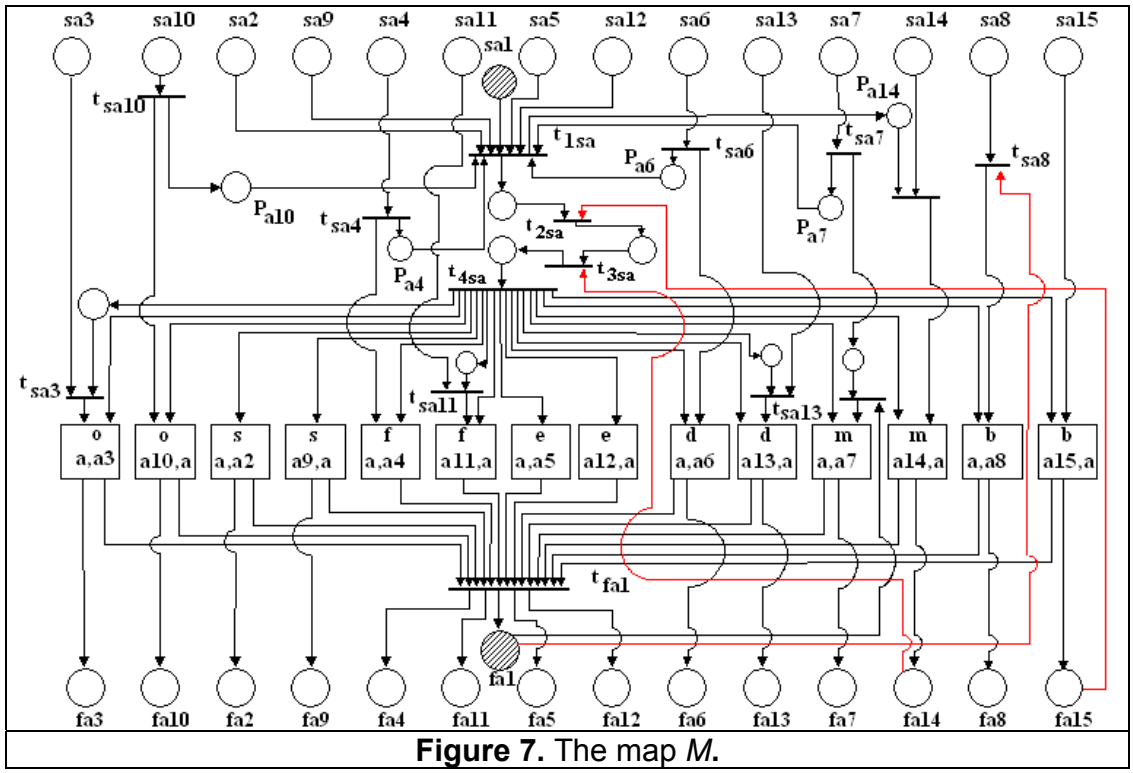


Figure 7. The map *M*.

In *M*, places sa_i indicate that task a_i is ready to start, and places fa_i indicate the end of a_i. The firing of transition tsa_i authorizes the beginning of a_i.

Transition t_{1sa} models conditions c1, c2, c7, and c8 (Section 4.1.1). The arc $(sa7,t_{1sa})$ indicates that *a7* is ready to start (part of condition c11). The arc $(t_{1sa}, Pa14)$ ensures that

a14 will start only when *a* is ready to start (part of c12). Transitions t_{sa10}, t_{sa4} and t_{sa6} respectively indicate to *a* that *a10*, *a4* and *a6* have already started (conditions c4, c5 and c9).

Transition t_{2sa} implements condition c14, and transition t_{3sa} implements the second part of c12. The firing of t_{2sa} must occur before the firing of t_{3sa} because task *a15* finishes before *a14*, as may be visualized in the consistency graph (Fig. 5). Tansition t_{4sa} authorizes simultaneously the beginning of task *a* at all LCM involved with it.

Transitions t_{sa3}, t_{sa11} and t_{sa13} respectively control the global conditions necessary for the beginning of *a3*, *a11*, and *a13* (conditions c3, c6 and c10, respectively).

Transition t_{sa7} authorizes the beginning of *a7* as soon as *a* finishes. (second part of condition c11). Transition t_{sa8} authorizes the beginning of *a8* if *a* has already finished (condition c13). Finally, transition t_{fa1} synchronizes the end of *a* with the end of *a4*, *a5*, *a11*, and *a12*.

The map *M* considers one instance of each possible relation to task *a*. However, two situations may occur: i) *a* does not have all of the 14 relations, ii) *a* is related with different tasks, but with the same kind of relation. In the first situation, the GCM is obtained by reproducing map *M* only with the existent relations. In the second situation, it is necessary to repeat the implementation of the conditions pertinent to the primitive that appears more than once. Fig. 8 (a) and (b) shows the GCMs of the stars *t3* and *t11* of the expression presented in Fig. 2.

Finally, it is necessary to consider the connection two adjacent stars. Suppose that a star *a* is generated in iteration *i* and a star *b* is generated in iteration *i+1*. Therefore, the relation connecting both stars (i.e., *r(a,b)* or *r(b,a)*) is modeled in the GCM of *a* and, following the schema of Fig. 6, there is a transition t_b that models the GC of *b* in relation to the tasks of star *a*. Similarly, there is in the GCM of *b* another transition t_b that models the GCs of *b* in relation to the tasks of star *b*. These transitions t_b, when fired, authorize the execution of *b*. Hence, to connect both stars, it is necessary to merge transitions t_b. This is done by eliminating t_b from the GCM of *a* and redirecting the GCs to t_b of the GCM of *b*, creating a single t_b that authorizes the LCM to execute *b*. Fig. 8 (c) shows the GCM resulting from the connection of stars *t3* and *t11*, and (d) shows the whole GCM for the expression of Fig. 2.

5. Conclusion

We present a methodology to describe and coordinate interdependencies of sets of tasks in collaborative environments. Based on the description, the algorithm to obtain the coordination mechanisms is shown.

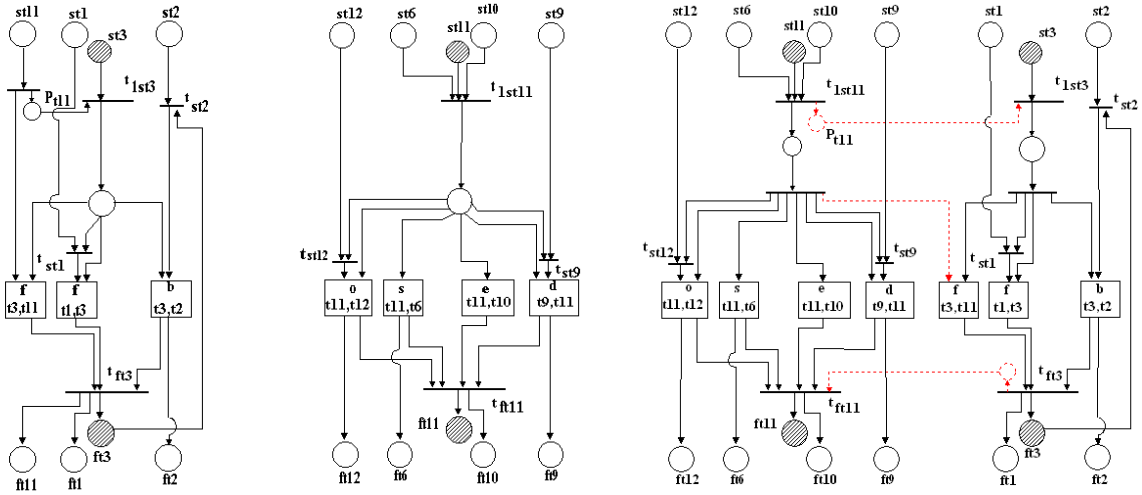
Among the contributions of this methodology, we can cite the global treatment given to the coordination process, i.e., a task is executed only if no relation of interdependency is violated.

Another approach to group coordination based on partial knowledge of the tasks structure of the environment is presented in [2]. Its goal is to create a framework for the coordination of a group (involving human and software agents) to guarantee that the tasks are completely solved in a timely and efficient manner.

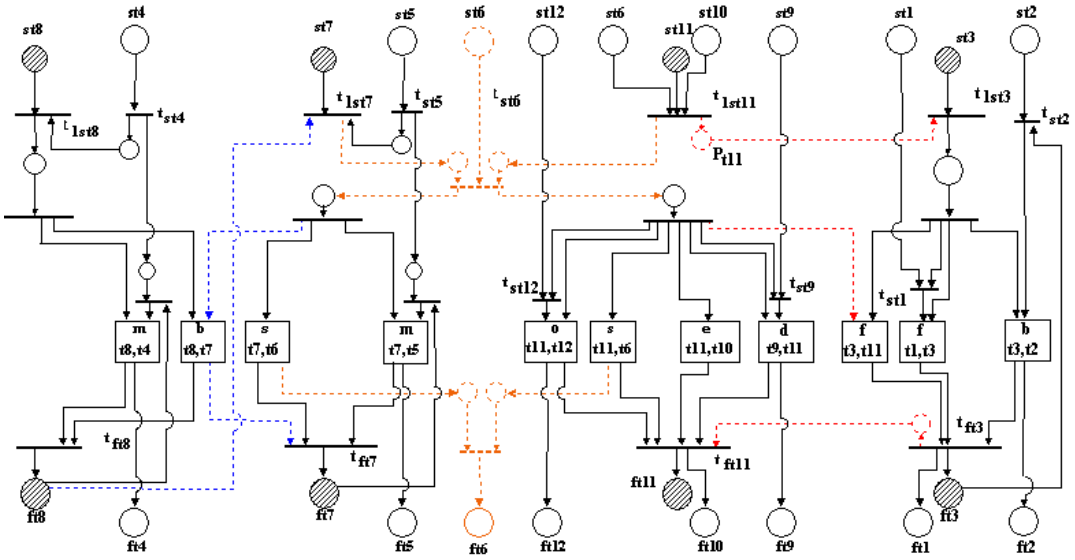
The use of concepts from the graph theory allowed an effective approach in the modeling of the global conditions, generating a standard procedure that is independent of the number of tasks taking part in the collaborative activity. This is a further step in previous

approaches that limit the number of tasks to which a task may be related (e.g., [5]).

The consistency graph used for the identification of the global conditions may also be used to verify if a set of tasks is realizable, i.e., if it does not generate an inconsistent configuration.



(a) GCM of t3 **(b) GCM** of t11 **(c)** Connection of GCMs of t3 and t11



(d) Compete GCM for the expression of Figure 2.

Figure 8. The construction of a GCM.

The relations graph offers a map of the whole collaborative activity and this, in conjunction with techniques from the graph theory, may be used for analysis processes in order to decide for example what will happen if a certain task is not executed.

Finally, it is important to reinforce that not all cycles in the relations graph generate inconsistencies. One of the next steps of this work is to investigate properties that could

enable the existence of cycles in the graph. Another future work is to create software components that implement the GCMs, based on the LCM implementation presented in [6].

6. References

[1] J. F. Allen, "Towards a General Theory of Action and Time", *Artificial Intelligence*, 23, 1984, pp. 123-154.

[2] K. S. Decker, "Coordinating Human and Computer Agents", In W. Conen, and G. Neumann (eds.), *Coordination Technology for Collaborative Applications – Organizations, Processes, and Agents*, LNCS 1364, Springer-Verlag, 1998, pp. 77-98.

[3] T. W. Malone, and K. Crowston, "What is Coordination Theory and How Can It Help Design Cooperative Work Systems?", *Conf. on Computer-Supported Cooperative Work (CSCW'90)*, 1990, pp. 357-370.

[4] A. B. Raposo, L. P. Magalhães, and I. L. M. Ricarte, "Petri Nets Based Coordination Mechanisms for Multi-Workflow Environments", *Int. J. Computer Systems Science & Engineering*, 15(5), .2000, pp. 315-326.

[5] A. B. Raposo, L. P. Magalhães, I. L. M. Ricarte, and H. Fuks, "Coordination of Collaborative Activities: A Framework for the Definition of Tasks Interdependencies", *7th Intl. Workshop on Groupware (CRIWG'2001)*, 2001, pp. 170-179.

[6] A. B. Raposo, A. J. Cruz, C. Adriano, and L. P. Magalhães, "Coordination Components for Collaborative Virtual Environ-ents", *Computers & Graphics*, 25(6), 2001, pp. 1025-1039.

[7] C. Simone, and K. Schmidt, "Taking the distributed nature of cooperative work seriously", *6th Euromicro Workshop on Parallel and Distributed Processing*, 1998, pp. 295-301.

[8] J. L. Szwarcfiter, *Grafos e algoritmos computacionais*, Editora Campus, Rio de Janeiro, 1986.

[9] A. K. Zaidi, "On Temporal Logic Programming Using Petri Nets", *IEEE Trans. Systems, Man, and Cybernetics – Part A: Systems and Humans*, 29(3), 1999, pp. 245-254.

Coordination Components for Collaborative Virtual Environments

Alberto B. Raposo[1,*] , Adailton J. A. da Cruz[1,2],
Christian M. Adriano[1], Léo P. Magalhães[1]

[1]Department of Computer Engineering and Industrial Automation (DCA)
School of Electrical and Computer Engineering (FEEC)
State University of Campinas (UNICAMP)
CP 6101 – 13083-970 – Campinas, SP, Brazil
Tel: +55-19-3788-3720 Fax: +55-19-3289-1395
{alberto, ajcruz, medeiros, leopini}@dca.fee.unicamp.br

[2] Universitary Center of Dourados – Federal University of Mato Grosso do Sul
CP 322 – 79804-970 – Dourados, MS, Brazil
Tel/Fax: +55-67-422-7118

Abstract.

This paper deals with the behavior of virtual environments from a the collaboration point-of-view, in which actors (human or virtual beings) interact and collaborate by means of interdependent tasks. In this sense, actors may realize tasks that are dependent on tasks performed by other actors, while the interdependencies between tasks (through resource management and temporal relations) delineate the overall behavior of a virtual environment. Our main goal is to propose an approach for the coordination of those behaviors. Initially a generic study of possible interdependencies between collaborative tasks is presented, followed by the formal modeling (using Petri Nets) of coordination mechanisms for those dependencies. In order to implement such mechanisms, an architecture of reusable and pluggable coordination components is also introduced. These components are used in an implementation of a multi-user videogame. The presented approach is a concrete step to create virtual societies of actors that collaborate to reach common goals without the risk of getting involved in conflicting or repetitive tasks.

Keywords. Modeling of behavior, collaborative virtual environment, coordination, computer-supported cooperative work, software components.

* Corresponding author.

1. Introduction

We are currently witnessing the rapid establishment of virtual societies, in which remote interaction is a feasible alternative to face-to-face contact, transcending geographic location and time constraints (“anywhere, anytime”) [13]. Two of the technological driving

forces in the virtual societies are computer-supported cooperative work (CSCW) and networked virtual environments (net-VEs).

CSCW is defined as "an endeavor to understand the nature and characteristics of cooperative work with the objective of designing adequate computer-based technologies [to support this kind of activity]" [4]. In other words, CSCW is interested in creating systems to support groups of people engaged in tasks with a common goal (i.e., collaboration).

A net-VE is a simulation of a real or imaginary world where multiple users may interact with one another in "real-time", share information, and manipulate objects in the shared environment [11, 27]. Net-VEs surpass the desktop metaphor of most current applications, proposing virtual communities where interactions are modeled according to the interactions in the real world.

Because of their great potential as tools for CSCW, net-VEs have been developed with an eye on CSCW results. This is particularly true for aspects such as user awareness, user embodiment and spatial models of interaction, for which results from the CSCW field have been successfully implemented in net-VEs [5, 10, 24]. When the net-VEs are aimed at collaborative activities, they are also called collaborative virtual environments (CVEs).

In spite of CVEs' suitable characteristics, there is still a gap between these environments and CSCW regarding the coordination of their activities. The development of CVEs has been dominated by leisure activities, basically enabling navigation through virtual scenarios and communication with remote users [8]. This kind of activity is what we call "loosely coupled collaborative activity" and is well coordinated by a social protocol, characterized by the absence of any explicit coordination mechanism, trusting the participants' abilities to mediate interactions.

On the other hand, "tightly coupled collaborative activities" require sophisticated coordination mechanisms in order to be efficiently performed in CVEs. In this kind of activity, tasks depend on one another to start, to be performed, and/or to end . A very illustrative example comes from the field of flight control and its tightly coupled activities of coordinating air traffic. Several technologies are under test to replace flight controllers' protocols for fully computer-based coordination solutions [21]. Other examples of tightly coupled activities may be found in collaborative authoring, workflow procedures, and multi-user computer games. Interdependencies among tasks in this kind of activity are normally positive, in the sense that each actor wants the others to succeed. However, they are not always harmonious. There must be coordination between tasks in order to ensure collaboration effectivenes. Without coordination, there is a risk that actors may get involved in conflicting or repetitive tasks. Coordination, in this context, is defined as "the act of managing interdependencies between activities performed to achieve a goal" [18] and is enacted by coordination mechanisms [26] that are software devices that interact with the application to control the behavior of a virtual environment.

The work presented in this paper is a step towards shortening the gap between CVEs and CSCW, by the introduction of an architecture of coordination components that may be used to control tightly coupled collaborative activities performed by means of CVEs. These components coordinate a set of interdependencies that often takes place between collaborative tasks, ensuring that these dependencies will not be violated. By using pluggable coordination components, different behaviors may be applied to the same CVE, just by changing components. Moreover, the components are generic and may be reused in other CVEs.

The following section overviews some related work. In Section 3, our coordination approach is presented. The approach is divided into three parts. Initially a generic set of interdependencies is defined. Then, the formal modeling of the coordination mechanisms is presented, using Petri Net as modeling and simulation tool. Finally, the coordination components architecture is depicted and its integration with the virtual environment is thoroughly discussed. Section 4 exemplifies the approach by means of implementing a multi-user videogame. Conclusions and suggestions for future research are drawn in Section 5.

2. Related work

This paper comprises three distinct areas of study, namely, CVEs, coordination (from the CSCW point-of-view) and software components. For this reason, we initially present a separate overview of related aspects of each of those areas and then explain how they are joined together in our approach.

2.1. Virtual Environments

Net-VEs are not a recent technology. Experimental systems have been around for decades, but only recently did they start to gain ground outside academic and military units. This popularity increase is mainly due to the rapid development of computing power and networking technologies, as well as their costs reduction.

In parallel to their growing popularity, a number of challenges remain to be overcome. Amongst them, we could mention all the problems related to management of network resources (concurrency, data loss, network failure, scalability, etc.); all the problems related to real-time graphics applications (e.g., CPU allocation for rendering); and those related to interactive multi-user applications (real-time data I/O, consistency among users, etc.) [27]. Additionally, there are also specific problems related to the application field of the net-VE, such as integration with large databases (e.g., geographic information about a terrain for military training environments); persistent storage (e.g., for engineering applications); and user authentication (e.g., for commerce applications).

When the application field is specifically collaboration, all the challenges related to CSCW should be also added to the challenges above. Just to name a few of them, there is the difficulty in relating spatial considerations to social interaction [5]; the difficulty in assisting individuals in working flexibly with virtual workplace objects [12]; and the necessity to create realistic avatars to improve communication among participants and their sense of presence [15].

The design of a CVE encompasses three distinct but interrelated aspects, namely, *form*, *function* and *behavior* [16]. Form is related to objects appearance, structure and physical properties, as well as the scene structure of the virtual world. Function refers to what objects do to accomplish their behavior (i.e., the actions they execute in the virtual world). Behavior refers to how objects define and dynamically change the different functions that they carry out over a period of time.

In this paper, we have been specifically concerned with behavioral aspects of CVEs. Although function and behavior are deeply related, the coordination approach to be presented clearly separates tasks (function) from task interdependencies and coordination

components (behavior). One of the advantages of this approach is the possibility of altering behaviors by simply altering the coordination components without having to alter the core of the CVE. For example, when a certain event takes place, the behavior of an actor might define in which direction it should walk . This behavior may be altered to define the direction in which the actor should run from then on, when the same event occur again. The executions of the tasks run and walk are independent from the coordination system, being related only to the form (e.g., the walk of a biped actor is different from that of a quadruped one).

2.2. Coordination in CSCW

Only in the second half of the 80's have collaborative systems with some kinds of coordination mechanisms started to appear. However, they were restricted to specific scenarios, because coordination protocols were rigidly defined, restraining dynamic modifications. The evidence that collaborative systems should not impose rigid work or communication patterns led to the development of systems that allow dynamic redefinitions and temporary modifications on the coordination model.

The idea of creating a set of tasks interdependencies and respective coordination mechanisms was proposed in the coordination theory of Malone and Crowston [18]. They defined three types of elementary resource-based dependencies and worked with the hypothesis that all other dependencies could be defined as combinations or specializations of these basic types. One of the advantages of this approach is the possibility to alter coordination policies simply by altering the coordination mechanisms for the interdependencies. Additionally, interdependencies and their coordination mechanisms may be reused. It is possible to characterize different kinds of interdependencies and identify the coordination mechanisms to manage them, by creating a set of interdependencies and respective coordination mechanisms capable of encompassing a wide range of collaborative applications.

Another use of interdependencies lies in the management of workflow activities [3]. In this case, interdependencies are defined as "constraints on the occurrence and temporal order of events", and are controlled by coordination mechanisms defined as finite state automata ensuring that they are not violated. The objective is to create a global scheduler that satisfies all dependencies defined for the workflow. A limitation of this work is that it is restricted to temporal interdependencies and is specific to workflow applications.

The present paper starts with some of the ideas of these previous works, and it is refined by defining a larger set of basic interdependencies that includes both temporal and resource management dependencies.

2.3. Software components

The first attempt to conceptualize the software components paradigm was directed by the concern with reuse and modularization [20]. Components were idealized based on an analogy with electronic systems, which are characterized by reusable modules with clear functions. This paradigm was forgotten for many years and then reappeared in a new software development context.

Nowadays, issues such as autonomy, interconnection and integration complement the component approach, in addition to reuse and modularization [28]. The paradigm advocates the idea that a component-based system should be composed of independent and autonomous parts compromised with their mutual decoupling (i.e., they are integrated in order to provide complete functions, but they do not either keep references to one another or communicate directly – the communication is achieved by means of messages). The decoupling compromise satisfies a number of requirements, such as the substitution of a component for another and the immediate insertion of new components (extensibility).

Regarding its fundamental elements, the component paradigm is defined by means of a software architecture [9], in which graph nodes are called components and the arcs are called connectors. Components are said to be active because they are responsible for all processing. Each component has a complete and self-contained service that is delivered through a specific interface pattern. By means of this pattern the component externalizes events and messages while still keeping its encapsulation. Connectors forward events and messages, and establish a loosely coupled integration between software components.

We have been implementing the component-based coordination architecture as a set of JavaBeans [14], which is a framework consisting of a Java API that enables the implementation of software components by the rules stated for the architectural style mentioned above. Software components are named beans, and connectors are objects of classes that extend appropriate event listeners.

For all the reasons discussed above, the component model is a very satisfactory solution for the design of CVEs, especially those supporting a task/interdependency model for tightly coupled collaborative activities.

As to the components' design, our choice for JavaBeans instead of other frameworks (e.g., DCOM) is motivated by the facility for integrating the Java language with Web-based CVEs.

2.4. Components as coordination mechanisms for CVEs

Our main goal is to create facilities for the design and implementation of CVEs. As already mentioned, coordination mechanisms are important in the use of CVEs for executing tightly coupled collaborative activities. The implementation of these mechanisms as components allows for the creation of an application-independent library of coordination components that permits incremental modifications to virtual environments. The application of component-based architectures in CSCW is a recent approach. One of the few works on this topic [25] extends the JavaBeans architecture to create a framework for building collaborative infrastructures.

Another important aspect of our approach is the development of a formal modeling of collaborative environments [23]. We use a Petri Net (PN) based modeling that enables the designer to anticipate and test the behavior of CVEs before their implementation, avoiding the trial and error approach. The choice for PNs as the modeling tool is justified because they are a well-established theory (there are numerous applications and techniques available) and can capture the main features of CVEs, such as non-determinism, concurrency and synchronization of asynchronous processes. Moreover, PNs accommodate models at different abstraction levels and are amenable both to simulation and formal verification. There are work results that also use PNs for modeling, analysis and simulation

of net-VEs [19, 29], however they are more related to physical aspects of CVEs implementation, such as tracking user's position and displaying the walls in an immersive environment. To our knowledge, there is no other work that uses PNs for simulation and analysis of actor's behaviors in CVEs (this idea has originated from a previous approach that uses PNs to model computer animations [17]).

3. The coordination approach

Before presenting our coordination model, it is necessary to clarify the definition of task used in this paper. In the present context, tasks are the "building blocks" of a collaborative activity, which is defined as a coordinated set of tasks performed by multiple actors to achieve a common goal. Tasks may be atomic or composed of subtasks and are connected to one another through interdependencies, and the management of such interdependencies is effected by coordination mechanisms. The granularity of a task reflects the interdependencies the current task has with other tasks. A group of subtasks with no external interdependencies (i.e., interdependencies with another task that does not belong to this group) may be nestled as a single task. For example, in a lower abstraction level, the walk of a bipedal structure may be considered a collaborative activity. In this activity, each limb may be considered an actor whose tasks are the desired movements that have interdependencies with the movements of other limbs (e.g., both legs may not be out of the ground at the same time). On the other hand, in a higher abstraction level, the walk may be considered a single task of a bipedal actor engaged in a more complex collaborative activity (e.g., a videogame).

Using this flexible definition of task, it is possible to model collaborative activities in several abstraction levels, which improves both the understandability and the feasibility of the interacting rules that characterizes the whole process.

The present section has three parts, namely, the definition of a set of frequent interdependencies between cooperative tasks; the formal modeling of coordination mechanisms to control these interdependencies; and the development of a library of coordination components to implement the modeled mechanisms.

3.1. Interdependencies

Interdependencies are divided into two types, *temporal* and *resource management*. This separation agrees with the coordination model proposed by Ellis and Wainer [7]. According to their model, the coordination in collaborative systems could occur in two levels, activity level and object level. At the activity level, the coordination model refers to temporal dependencies, describing "the sequencing of activities [tasks] that make up a procedure [collaborative activity]." At the object level, the coordination model refers to resource management dependencies, describing "how the system deals with multiple participants' sequential or simultaneous access to some set of objects."

3.1.1. Temporal interdependencies

Temporal interdependencies establish the execution order for tasks. The set of temporal interdependencies of our coordination model is based on temporal relations defined by Allen [2]. According to him, there is a set of primitive and mutually exclusive relations that could be applied over time intervals (and not over time instants). This characteristic made these relations suited to task coordination purposes, because tasks are generally non-instantaneous operations.

The temporal logic of Allen is defined in a context where it is essential to have properties such as; the definition of a minimal set of basic relations; the mutual exclusion among these relations; and the possibility of making inferences over them. Temporal interdependencies between collaborative tasks, on the other hand, are inserted in a different context. For this reason, it was necessary to make some adaptations to Allen's basic relations, adding a couple of new relations and creating some variations of those originally proposed. The main difference in the context of collaborative activities is that it is possible to relax some restrictions imposed by the original relations. This introduces a degree of redundancy from a temporal logic's point-of-view, but makes the coordination model more manageable. The result of the adaptation of Allen's relations tothe context of collaborative activities is the set of 13 temporal interdependencies presented below [22].

Considering two tasks T1 and T2 that occur, respectively, in time intervals $[t1_i, t1_f)$ and $[t2_i, t2_f)$,

T1 **equals** T2 (t1i = t2i and t1f = t2f): This dependency establishes that two tasks must occur simultaneously.

T1 starts T2: This relation has been divided into two.

- **T1 startsA T2** ($t1_i = t2_i$ and $t1_f < t2_f$): Both tasks must start together and the first must end before. This is the original relation proposed by Allen.
- **T1 startsB T2** ($t1_i = t2_i$): Variation of the original relation, relaxing the obligation of the first task having to end first. This variation makes sense because in some situations, it is required that both tasks start together, but it does not matter when they end.

T1 finishes T2: Similarly to the previous one, it is possible to define two relations based on it.

- **T1 finishesA T2** ($t1_i > t2_i$ and $t1_f = t2_f$): Both tasks end together, but the first must start after the second. This is the original relation.
- **T1 finishesB T2** ($t1_f = t2_f$): Similarly to startsB, this dependency is obtained from the original, relaxing the restriction that T1 must start after T2. This dependency is important for situations in which it does not matter when tasks have begun, as long as they end simultaneously.

T1 before T2: This relation clearly illustrates the difference between Allen's temporal logic and task interdependencies. It can be divided into three distinct interdependencies.

- **T2 afterA T1** ($t1_{f,n} < t2_{i,n}$, $\forall n > 0$, where n means the n^{th} execution of the task): T2 may only be executed if T1 has already ended (the restriction occurs in the execution of T2; T1 can be freely executed). This dependency is the prerequisite relation, which is very common in collaborative applications. In this case, T2 may be executed only once after each execution of T1.

T2 afterB T1 ($t1_{f,1} < t2_{i,n}$, $\forall n > 0$): Variation of the previous dependency, in which T2 may be executed several times after a single execution of T1.

T1 beforeA T2 ($t1_f < t2_i$): From a temporal logic point-of-view, this relation is the opposite to after (the formal definition is the same). However, they generate totally different coordination mechanisms. Essentially, the difference is that in this case the restriction occurs in the execution of T1, which may not be further executed if T2 has already started its execution. There is no restriction to the execution of T2 (T2 does not have to wait for the execution of T1, as it would happen to the dependency T2 afterA T1).

T1 meets T2 ($t1_f = t2_i$): T2 must start immediately after the end of T1.

T1 overlaps T2: This relation is divided into two types.

T1 overlapsA T2 ($t1_i < t2_i < t1_f < t2_f$): T1 starts before T2, and T2 must start before the end of T1, which must end before T2. It is the original relation.

T1 overlapsB T2 ($t1_i < t2_i < t1_f$): Variation of the original relation, in which it does not matter which task ends first. The only obligations are that T1 starts before T2, and T2 starts before the end of T1.

T1 during T2: This relation is also adapted to generate two new interdependencies.

T1 duringA T2 ($t1_{i,n} > t2_{i,n}$ and $t1_{f,n} < t2_{f,n}$, $\forall n > 0$): T1 must be totally executed during the execution of T2. In this case, a single execution of T1 is allowed during an execution of T2.

T1 duringB T2 ($t1_{i,n} > t2_{i,m}$ and $t1_{f,n} < t2_{f,m}$, $\forall m > 0$ and $\forall (n \geq m)$): Variation of the previous dependency, in which T1 may be executed more than once during a single execution of T2.

A consequence of the included redundancies in Allen's logic is that there is not a unique way to represent interdependencies among tasks, but these redundancies give a more understandable and manageable perspective to the model.

3.1.2. Resource management interdependencies

Resource management interdependencies are complementary to temporal ones and may be used in parallel to them. This kind of interdependency deals with the distribution of resources among tasks. Three basic resource management dependencies are defined.

Sharing: A limited number of resources may be shared among several tasks. It represents a common situation that occurs, for example, when several users want to edit a document.

Simultaneity: A resource is available only if a certain number of tasks request it simultaneously. It represents, for instance, a machine that may only be used with more than one operator.

Volatility: Indicates whether, after its use, the resource is available again. For example, a printer is a non-volatile resource, while a sheet of paper is volatile.

From the basic interdependencies above, it is also possible to define composite interdependencies. For example, sharing M + volatility N indicates that up to M tasks may share a resource, which may be used N times only.

Different from temporal dependencies, resource management dependencies are not binary relations. It is possible, for example, that more than two tasks share a resource. Moreover, each of the above interdependencies requires parameters indicating the number

of resources to be shared; the number of tasks that must request a resource simultaneously; and/or the number of times a resource may be used (volatility).

3.2. Modeling coordination mechanisms

The mechanisms to coordinate the above set of interdependencies were modeled by using PNs. The formal modeling of the mechanisms not only does enable a previous verification and validation of the CVE's behavior, but also constitutes the internal logic of the coordination components (which will be presented in the next section).

In the proposed scheme, the design of a collaborative environment is divided into three distinct hierarchical levels, workflow*,* coordination *and* execution *(Figure 1).*

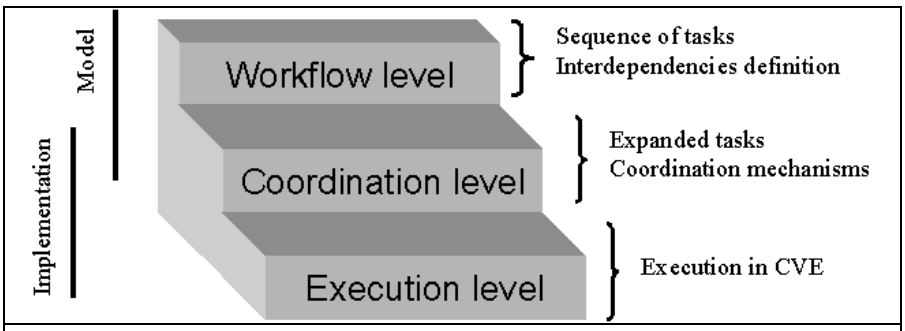


Figure 1: Hierarchical levels for the design of CVEs.

At the workflow level, a global sketch of the environment's behavior is delineated, establishing which elements of the scene will be considered actors and which tasks are assigned to them. Each actor's behavior is modeled separately, establishing the interdependencies between tasks of the same actor or those of different actors.

The coordination level is built under the workflow level by the expansion of interdependent tasks according to a PN-based model defined in [1] and the insertion of correspondent coordination mechanisms between them. At this level, we have a complete PN model of the actors' behaviors, allowing anticipation of undesired situations (such as deadlocks) through simulation, verification, validation and performance analysis of the model.

The execution level deals with the actual execution of tasks in the CVE. This level is the interface between the coordination infrastructure and the CVE. It will be treated in the next section.

During the passage from the workflow to the coordination level, each task that has a dependency on another is modeled by a system with five transitions (ta, tb, ti, tf and tc) and four places (P1, P2, P3 and P4), as proposed by van der Aalst et al. [1]. As shown in Figure 2, attached to each expanded task, there are also five places that represent the interaction with the resource manager, the temporal coordination mechanism, and the agent that executes the task. The places request_resource, assigned_resource and release_resource connect the task with the resource manager. The places start_task and finish_task connect the task with the temporal coordination mechanism and the agent that performs it, respectively indicating the beginning and the end of the task execution.

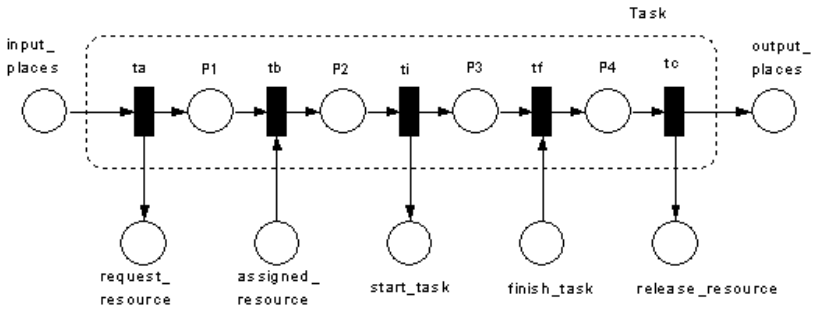


Figure 2: Task model in the coordination level.

In order to illustrate a temporal coordination mechanism, Figure 3 presents the model of the mechanism for relation Task 1 startsA Task 2. In this figure, t1 is a control transition that ensures the simultaneous beginning of both tasks, and t2 ensures that Task 2 will finish only after the end of the other task. The transitions called task1 and task2 represent the real execution of the tasks. They are modeled by means of transitions with token reservation (represented with the letter "R"), which are non-instantaneous transitions (tokens are removed from their input places when they fire and only some time later are they added to their output places, representing the duration of the tasks' execution).

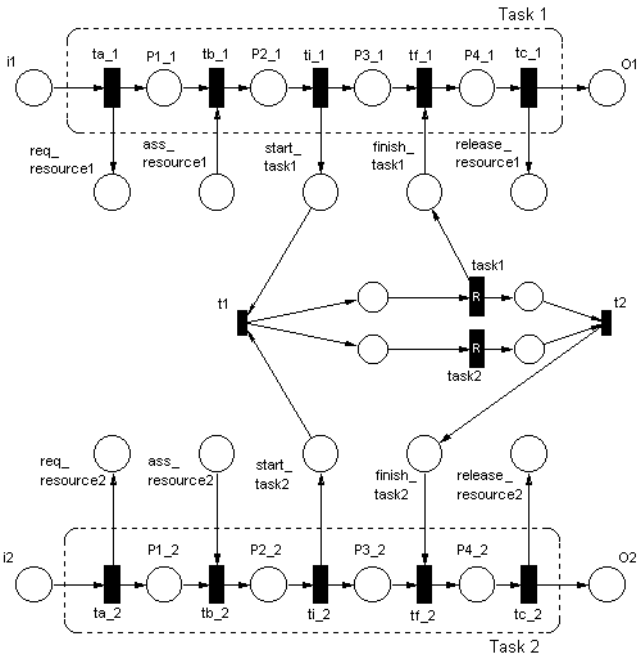


Figure 3: Model of the coordination mechanism for Task 1 startsA Task 2.

Another mechanism presented here is the one for managing the resource dependency sharing N. The coordination mechanism for this dependency consists of a place Pn with N tokens representing the available resources. This place is the input place for a transition

connecting request_resource to assigned_resource, defining whether there are available resources. At the end of the task, the token returns to Pn via release_resource. Figure 4 shows the model for N = 3.

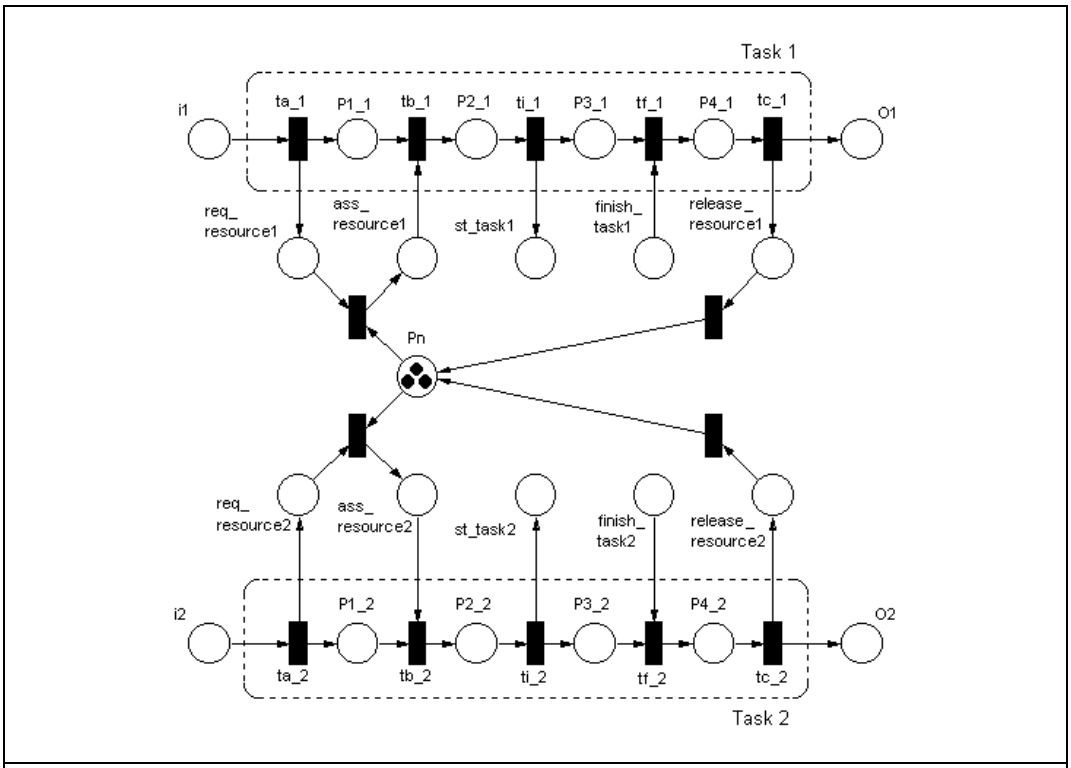


Figure 4: Model of the coordination mechanism for two tasks sharing 3 resources.

A detailed explanation of the coordination mechanism models has been presented elsewhere [22, 23]. The full set of models is available at http://www.dca.fee.unicamp.br/~alberto/pubs/IJCSSE/mechanisms.

The next step is to create software components that implement the modeled coordination mechanisms as "black boxes" connecting the collaborative tasks. The coordination components are capable of receiving and generating events from/to tasks in the CVE, controlling their execution. Thus, events related to tasks (e.g., the beginning of a task) may generate output events that affect the execution of interdependent tasks (e.g., blocking the execution of another task).

3.3. Coordination components

We have idealized the coordination level as a software layer loosely coupled to the scene. The advantages of this approach are twofold. It isolates the scene from changes made exclusively on the coordination architecture, and also provides a clear test bench to investigate coordination issues. The list of requirements that should be supported by the loosely coupled coordination layer is as follows:

1. Implementation separated from the scene.

2. Accommodation of changes at the scene, such as removal or inclusion of a task.
3. Changes in the coordination logic should be possible and be restricted to a few software elements.
4. Mixing of different coordination logics within the same coordination control.
5. Attachment of monitoring tools to gather performance related to the coordination control strategy. Potential evaluation indices are throughput, deadlock situations, and load balance.

The architectural decoupling between components and the three-level architecture are solutions for requirements 1-3. The encapsulated implementation of complete services within each component is the solution for requirement 4. The event-oriented communication is the basis for attaching external tools (requirement 5).

3.3.1. Coordination Level

Figure 5 illustrates a high level vision of the components involved in the coordination of interdependencies between two tasks. At the coordination level the figure shows three components, a coordinator and two tasks. The coordinator component implements the coordination mechanisms, both for temporal and resource management dependencies. The task component, instantiated for each task, is responsible for maintaining the task's schedule.

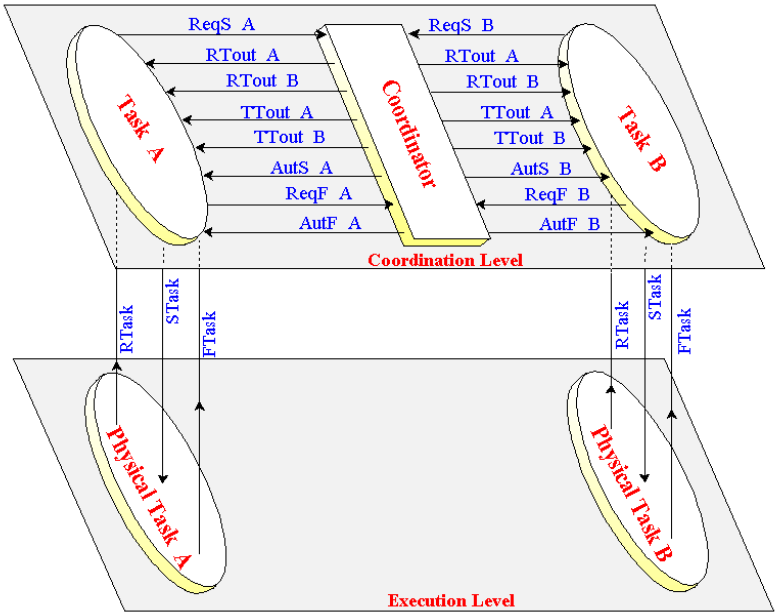


Figure 5: High-level vision of the coordination component between two tasks.

The components of our architecture were designed to communicate by means of the following sequence of events:

REQUEST START (ReqS): When a task is demanded, for example due to a user's action, it must first contact the coordinator to request an authorization for execution. At this time, the coordinator checks if there is any resource management interdependency and, if so, it consults the resource management mechanism to verify if the resource is available (if not, it has to wait until the resource becomes available). Once the resource is available or in the case of having no resource dependency, the coordinator checks if there is any temporal interdependency. If so, it consults the temporal coordination mechanism to verify if the conditions for the task to start have been satisfied (if not, it has to waituntil these conditions are satisfied). Once all conditions are satisfied, the signal AutS is sent to the task component.

AUTHORIZE START (AutS): This signal is the authorization given by the coordinator that enables the beginning of a task's execution.

REQUEST FINISH (ReqF): Once the task wants to end its execution, it sends this signal to the coordinator, which verifies if the temporal interdependency (whether it exists) enables the end of the task. If so, it sends the signal AutF to the task and, if there is a resource management dependency, it releases the assigned resource. Otherwise, the coordinator waits until the temporal coordination mechanism authorizes the end of the task.

AUTHORIZE FINISH (AutF): This signal indicates to the task that it may end.

In the model of the coordinator component, a task has to wait until the temporal and the resource management coordination mechanisms authorize its execution. A possible consequence of this fact is that the task may wait indefinitely if either of these conditions is not satisfied. In order to avoid such situations, the coordinator also sends timeout signals to the tasks.

RESOURCE TIMEOUT (RTout): If the resource is not assigned to the task after a certain waiting time, the coordinator sends this signal.

TEMPORAL TIMEOUT (TTout): If another task does not offer the conditions for the beginning of the task that has requested it, the coordinator sends this signal after a certain waiting time. In this case, if the resource has already been assigned to the task, the coordinator also releases it.

The treatment of timeouts is left to the task components. This gives more flexibility to the architecture, because it does not risk the coordinator component reusability.

The task that had its start unauthorized due to a timeout may execute an alternative task when it receives that signal. This kind of timeout can be thought as an "emergency procedure" to prevent other tasks from being blocked. For example, in a virtual laboratory, the students must follow a sequence of experiments in order to achieve the expected skills. However, if a student cannot conclude a certain experiment after a pre-defined time, a virtual helping agent could be activated in order to guide the user through all the steps of the experiment.

Another possible timeout treatment is to return the task to its initial state. This approach only works when the collaborative activity has an alternative path that avoids the blocked task. In a virtual chat room, for example, if a user cannot start a conversation because there is nobody available, he/she could return to the virtual hall and choose another room.

Another possible consequence of timeouts is that the non-execution of an expected task may invalidate interdependent tasks previously executed. For this reason, the coordinator component sends the timeout signals to all interdependent tasks, and not only to that one which had its execution unauthorized. An example occurs in relation Task2 afterA Task1. Task1 may be executed without restrictions, but in some cases it expects the execution of Task2 to be "validated." This situation occurs, for example, in e-commerce, where the processing of an order must be followed by the payment. If the payment is not effectuated, after a certain period the order should be canceled. In the example to be presented in the next section, this situation occurs if the monster's wounds are not cauterized after a certain time its head has been severed. In this case, the head is regenerated and the hero must sever it again to kill the monster.

The coordination component encapsulates two PN simulators, one for the temporal and another for the resource management coordination mechanism. Each of these simulators interacts with their associated events, received or sent by the component. Events arriving at the coordinator alter the state of the simulator, while certain states of the simulator may generate output events. Figure 6 sketches the coordinator functioning.

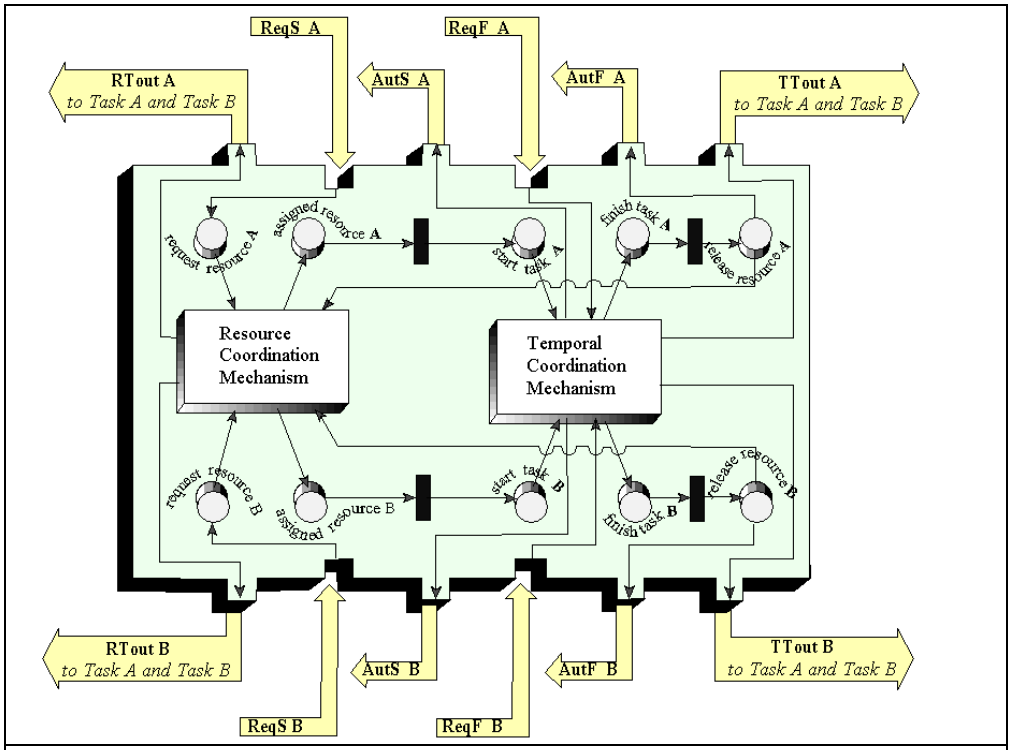


Figure 6: The coordinator component functioning.

When the coordinator receives a ReqS signal, it puts a token in the respective request_resource place, starting the resource management mechanism. When the resource is available, the resource management mechanism puts a token in the respective assigned_resource place, which sends it to start_task. The start_task place starts the temporal coordination mechanism, which sends the respective AutS signal indicating the

time the task may begin. When the temporal mechanism receives the ReqF signal, it checks whether the logical end of the task is authorized and, if so, sends a token to finish_task, which is then passed on to release_resource, indicating to the resource management mechanism that the resource is free again. Finally, the coordinator sends the AutF signal, indicating the logical end of the task.

Although belonging to the same component, each internal coordination mechanism has independent behavior, running a different PN simulator. A natural design choice would have been to consider each mechanism a component itself. However, this would cause a communication overhead between both components when a pair of tasks had both kinds of interdependencies (i.e., temporal and resource management), harming the decoupling characteristic of the components. The choice for considering a component as composed of two coordination mechanisms does not reduce the flexibility of the model, because the component is customizable, i.e., the kind of interdependency to be treated by each of the mechanisms is a parameter of the component. In the particular situation where there is only one kind of interdependency, the other internal mechanism may be considered "empty" with no effects in the coordination of tasks.

3.3.2. Execution Level

In the previous section we have centered the discussion on the coordination level, where only the logical execution of tasks is considered. The implementation of the coordination components must also consider the execution level, which represents the "physical" execution of the tasks in the CVEs. As shown in Figure 5, the connection between coordination and execution levels is left to the task components, reducing the responsibilities of the coordinator components, a feature that contributes to their reuse.

The physical tasks communicate with their respective task components by means of the following events:

REQUEST TASK (RTask): This signal is sent from the physical task to the respective component when an event in the CVE (user action, solicitation of another task, elapsed time, etc.) requests its start. The task component forwards this information to the coordinator at the coordination level via ReqS signal.

START TASK (STask): When the coordinator authorizes the start of a task (AutS), the task component forwards this authorization to the physical task via this signal. At this moment, the task really starts its execution in the CVE.

FINISH TASK (FTask): The end of the task's execution in the CVE generates this signal that is sent to the respective component at the coordination level. This information is forwarded to the coordinator via ReqF signal. It is important to clarify that the logical end of the task (i.e., the end from the coordination level point-of-view) occurs only when the task component receives the AutF signal from the coordinator. The logical end of a task guarantees that the temporal interdependencies will not be violated, but may be delayed in relation to its physical end in specific situations.

An advantage of this multiple-level approach is the modularization of the architecture. The coordination model of the CVE may be developed independently from its implementation and vice versa.

Although the only commitment between the coordination and execution levels is the communication by means of the three signals discussed above, the integration issue is not an easy task. This is worsened by the tight relation between scene semantics and its coordination, which directly impacts the communication pattern within both levels. The choice for a communication pattern imposes different kinds of externalization of scene internal elements (i.e., actors, resources, tasks, etc.). In the following we discuss three possibilities of externalization (Figure 7).

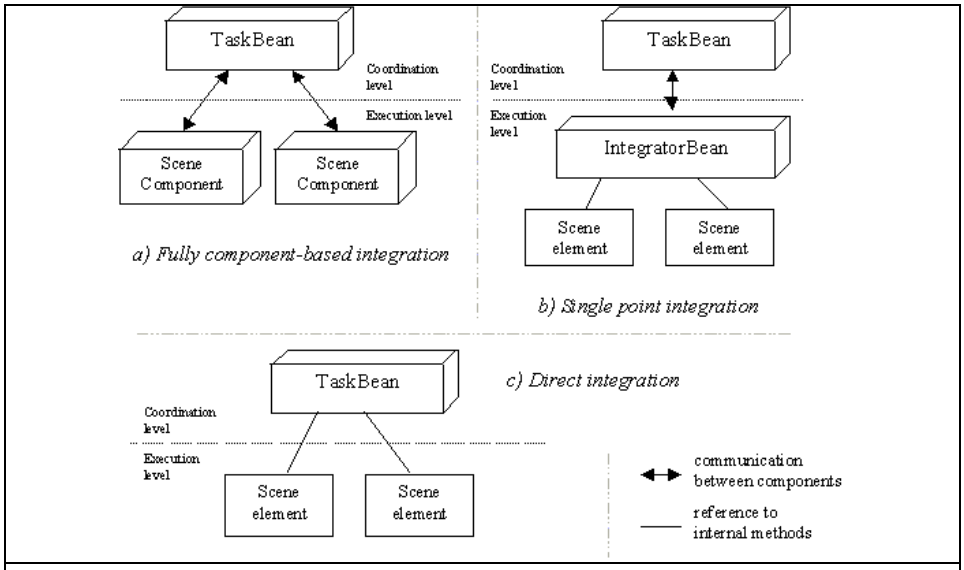


Figure 7: Approaches for the integration scene-coordination components.

The first possibility is to implement the CVE under the software component paradigm (i.e., scene objects, actors, tasks, and resources implemented as components), which would lead to a straightforward integration with the coordination control. Each task component at the coordination level would be connected to a set of components within the scene. In this approach connections are loosely coupled because the task component (TaskBean - Figure 7a) does not need to import any scene library/package. This fully component-based integration approach is the ideal one from the software component architecture point-of-view, but it requires a component-based CVE.

A second alternative would be to encapsulate the whole scene into a single IntegratorBean, which would be basically responsible for forwarding coordination commands to the appropriate scene elements and returning scene events to the respective TaskBeans at the coordination level (Figure 7b). This single point integration approach solution has the advantage of not imposing a reorganization of scene elements, enabling different scene technological implementations. Its drawback is a less decoupled and robust solution, since even a slight change made to the scene would impact the IntegratorBean.

A third approach is to have each TaskBean accessing their respective scene elements (Figure 7c). The advantage of this direct integration approach is that there is not a unique integration point and, therefore, no need to translate JavaBeans events to method calls in scene elements. Moreover, there is no need to restrict or adapt the scene to the

component model. Scene elements may even be implemented as simple structured (non object-oriented) code. The drawbacks, however, are numerous. The sense of layer is wicked, because coordination components need to import scene libraries. Furthermore, even a single alteration to the scene could impact the coordination components in a less uniform and localized manner than in previous integration approaches.

Since the implementation of a fully component-based CVE is out of the scope of this paper, and our initial goal is to use the coordination approach in existent CVEs without further modifications, we use the direct integration in the videogame implementation presented in the following.

4. Prototype implementation: A multi-user videogame

In this section a case study of a CVE is presented, where a user interacts with an autonomous agent that represents a second user. The example implements a kind of videogame based on the second "task" (activity) of Heracles, from the Greek mythology. According to the legend, Heracles had to kill the Hydra of Lerna, a monster with nine heads that are regenerated after being severed. In order to achieve his goal, Heracles needs the collaboration of his nephew Iolaus, who cauterizes the monster's wounds after Heracles cuts each head off. However, the last head may not be severed by any weapon. The solution is to bury the monster in a deep hole and cover it with a huge stone.

Figure 8 illustrates the PN model of the videogame (open rectangles indicate interdependent tasks). There are two identical nets, one representing the user's and the other representing the agent's sequence of tasks. Each net has two alternative paths, indicating that each "actor" (user or agent) may assume either role (Heracles or Iolaus). The upper part of the nets represents Heracles' sequence of tasks and the lower part, Iolaus' sequence of tasks. Heracles must get the sword, sever eight of the Hydra's heads, throw the beast into the hole and cover it with a stone. Iolaus, on the other hand, must get the torch, cauterize the wounds after Heracles *has severed the heads and dig the hole.*

At the coordination level, the boxes with interdependency names are replaced by the respective coordination mechanism models for simulation and analysis purposes. At the execution level, these boxes represent the internal mechanisms of the coordination components to be used.

The definition of which actor will assume which role is given by the interdependencies volatility 1 between the tasks get_sword and get_torch. Since there are only one sword and one torch available, the choice of weapon determines the role to be played by each actor. There is also an equals interdependency between get_sword and get_torch of different actors. This interdependency forces the agent to choose the other weapon when the user chooses his/hers.

The interdependency sharing 1 among the tasks get_sword, sever_head and cauterize is the "core" of the game. When Heracles gets the sword, he is also assigned eight resources that may be thought as "abstract authorizations" to cut Hydra's heads. After he has severed each head, a resource is released, indicating to Iolaus that he may cauterize that wound. If the head wound is not cauterized within a certain period after it has occurred, the

timeout signal sent by the coordination mechanism causes the task to return to its initial state and also reassigns the resource to Heracles, indicating that he must once again sever that head (now regenerated).

There is also an interdependency afterA, indicating that Heracles may only throw the monster in the hole if Iolaus has already dug it.

The PN model was simulated and analyzed with the aid of a developed tool [23] (actually, the model shown in Figure 8 is an enhancement of previous models whose simulation has detected problems).

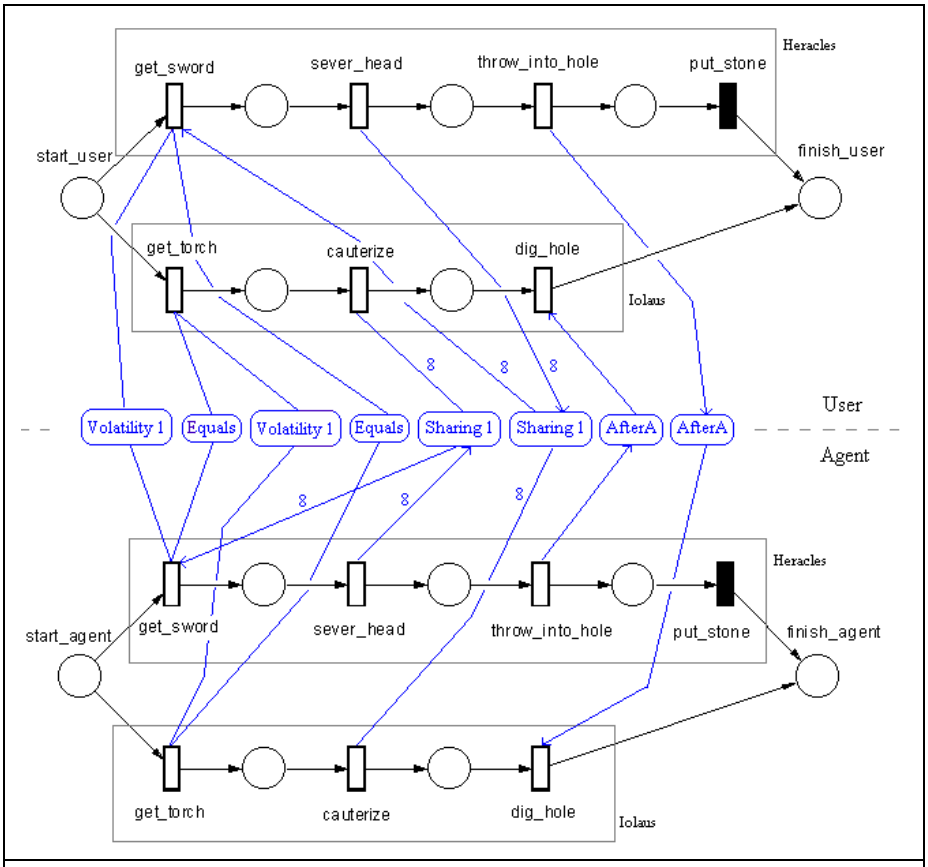


Figure 8: PN model of Heracles videogame.

The videogame was implemented by using the Blaxxun Contact [6], a client for multimedia communication that provides resources for VRML (Virtual Reality Modeling Language) visualization, chats, message boards, avatars, etc. In this implementation we have used the VRML visualization and the avatars with pre-defined movements.

The interaction with the user occurs by means of buttons defined in a Java applet that interacts with the VRML world via EAI (External Authoring Interface), an interface that enables external programs to interact with objects of a VRML scene. By clicking on the applet's buttons, the user orders the execution of a task in the virtual world (the physical execution of the tasks is decoupled from their coordination). At the implementation, task

components are connected to the interface's buttons, enabling or disabling them whether their respective tasks are enabled or not. The integration of the components with the VRML scene is direct (Figure 7c) because the components call methods of the applet graphical interface elements, and vice versa. The applet, however, could be redesigned as an integrator bean (Figure 7b), which would enhance the decoupling of the architecture.

In order to give more dynamism to the game, the agent has an aleatory behavior, taking a variable time to start the execution of the tasks imputed to it. For example, when the user assumes the role of Heracles, the agent (Iolaus) may not cauterize a head cut by Heracles before it is reborn. Figure 9 shows some frames of the videogame (the nine heads of Hydra are represented by nine monsters). Frames a and b show Heracles' interface, while frames c and d show Iolaus' interface.

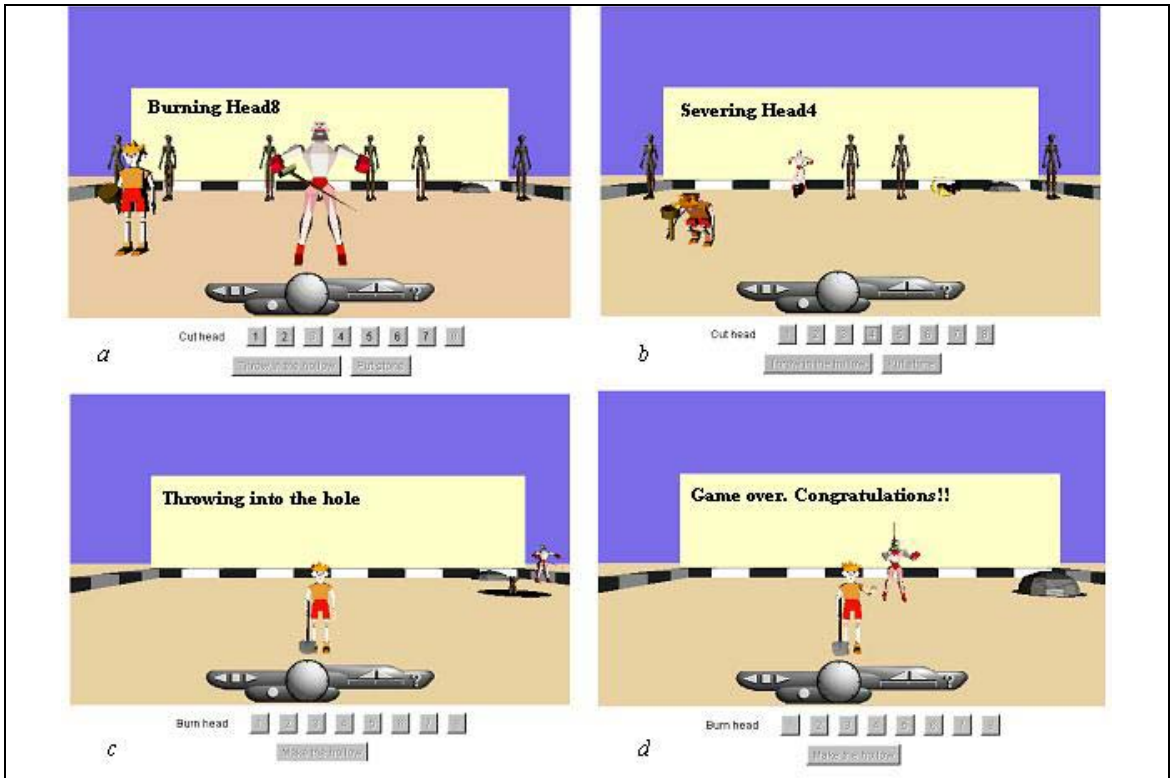


Figure 9: Frames of Heracles videogame.

5. Conclusion

This paper has presented an approach for behavior coordination in CVEs, which is a step towards shortening the gap between virtual environments and CSCW practices regarding the carrying-out of tightly coupled collaborative activities. The approach started with the basic idea of defining a set of interdependencies that frequently occur between collaborative tasks. For each interdependency, a coordination mechanism was formally defined as a PN model. At this level, the whole environment may be expanded as a PN for simulation and analysis, enabling the anticipation of possible problems. Finally, we have

also developed a component-based architecture to implement the coordination control in CVEs in a modular and pluggable way.

Our coordination approach follows a three-abstraction levels hierarchy, which has the advantage of isolating the parts of coordination design. For instance, if the CVE designer wants to implement a videogame such as the one presented in Section 4, it is not necessary to consider the PN model of the system. Designers may simply use pre-defined coordination components, which encapsulate the PN logic of the coordination mechanisms, hiding this logic from designers. Thus, it is not necessary to attach a PN model to a CVE. On the other hand, in initial phases of the design, only the model may be used, dismissing implementation details.

Another important aspect of the presented coordination approach is that it treats the CVE's behavior independently from its form and function. In order words, the behavior does not depend on animation and modeling techniques. For example, the videogame presented in Section 4 may be redesigned to include more sophisticated scenario, avatars and movements without afecting the coordination infrastructure. Moreover, a whole task execution may be redefined without affecting the global behavior (for instance, instead of using the sword to sever the heads, Heracles could get a gun to shoot them).

As future research, we plan to formalize the presented approach (from the coordination logic to components implementation) in order to create a generic coordination framework that can be used not only in CVEs, but also in other kinds of collaborative applications. Concerning the CVE application, we have developed here a basis for further research on the assumption that virtual environment behavior reflects collaboration patterns that are the outcome of coordination control strategies. Isolating the three dimensions is a fundamental step to understand how they intertwine in CVE. Following this reasoning, we suggest that an ample research on infrastructure for CVE is already necessary. The coordination apparatus demonstrated here is one of the many aspects of such complex framework.

Finally we would like to reinforce our belief that the upcoming virtual society will be strongly based on CSCW and net-VE technologies. For this reason, research ventures towards shortening the gaps between both technologies, such as that of coordination focused here, are important steps towards the settlement of this society.

Acknowledgements. The first author is sponsored by FAPESP (Foundation for Research Support of the State of São Paulo, Brazil), grant number 00/10247-3. The second author is sponsored by CAPES/PICD. Thanks also to DCA/FEEC/Unicamp and CEUD/UFMS for the expressive support granted to this research.

References

[1] van der Aalst WMP, van Hee KM, Houben GJ. Modelling and analysing workflow using a Petri-net based approach. Proceedings 2nd Workshop on Computer-Supported Cooperative Work, Petri nets and related formalisms, 1994. p. 31-50.

[2] Allen JF. Towards a General Theory of Action and Time. Artificial Intelligence 1984; 23: 123-154.

[3] Attie PC et al. Scheduling workflows by enforcing intertask dependencies. Distributed Systems Engineering Journal 1996; 3(4): 222-238.

[4] Bannon LJ, Schmidt K. CSCW: Four Characters in Search of a Context. In: Bowers JM, Benford SD, editors. Studies in Computer Supported Cooperative Work, North-Holland, 1991. p. 3-16.
[5] Benford S et al. Supporting Cooperative Work in Virtual Environments. The Computer Journal, 1994; 37(8): 653-668.
[6] blaxxun interactive. *blaxxun Contact 4.4*. <http://www.blaxxun.com/products/contact>, August 2000.
[7] Ellis CA, Wainer J. A Conceptual Model of Groupware. Proceedings ACM Conference on Computer Supported Cooperative Work (CSCW), 1994. p. 79-88.
[8] Frécon E, Nöu AA. Building Distributed Virtual Environments to Support Collaborative Work. Proceedings ACM Symposium on Virtual Reality Software and Technology (VRST), 1998. p. 105-119.
[9] Garlan D, Shaw M. Introduction to Software Architecture. In: Ambriola V, Tortola G, editors. Advances in Software Engineering and Knowledge Engineering, Vol. I, World Scientific Publishing Co., 1993.
[10] Gutwin C, Greenberg S. Workspace Awareness for Groupware. Proceedings ACM Conference on Human Factors in Computing Systems (CHI), 1996. p. 208-209.
[11] Hagsand, O. Interactive Multiuser VEs in the DIVE System. IEEE Multimedia 1996; 3(1): 30-39.
[12] Hindmarsh J et al. Object-Focused Interaction in Collaborative Virtual Environments. ACM Transactions on Computer-Human Interaction 2000; 7(4): 477-509.
[13] Igbaria M. The Driving Forces in the Virtual Society. Communications of the ACM 1999; 42(12): 64-70.
[14] Sun Microsystems. *JavaBeans*. <http://java.sun.com/products/ javabeans>, May 2001.
[15] Joslin C et al. Sharing Attractions on the Net with VPark. IEEE Computer Graphics and Applications 2001; 21(1): 61-71.
[16] Kim GJ et al. Software Engineering of Virtual Worlds. Proceedings ACM Symposium on Virtual Reality Software and Technology (VRST), 1998. p. 131-138.
[17] Magalhães LP, Raposo AB, Ricarte ILM. Animation Modeling with Petri Nets. Computers & Graphics 1998, 22(6): 735-743.
[18] Malone TW, Crowston K. What is Coordination Theory and How Can It Help Design Cooperative Work Systems? Proceedings ACM Conference on Computer-Supported Cooperative Work (CSCW), 1990. p. 357-370.
[19] Mascarenhas R et al. Modeling and Analysis of a Virtual Reality System with Time Petri Nets. Proceedings International Conference on Software Engineering (ICSE), 1998. p. 33-42.
[20] McIlroy MD. Mass Produced Software Components. In: Naur P, Randell B, editors. Software Engineering, Report on a conference sponsored by the NATO Science Committee, Scientific Affairs Division, NATO-Brussels, 1968. p. 138-155.
[21] Perry, TS. In search of the future of air traffic control. IEEE Spectrum 1997, 34(8):18-35.
[22] Raposo, AB. Coordination in Collaborative Environments Using Petri Nets. Doctorate Thesis, DCA - FEEC – UNICAMP, October 2000. In Portuguese.
[23] Raposo AB, Magalhães LP, Ricarte ILM. Petri Nets Based Coordination Mechanisms for Multi-Workflow Environments. International Journal of Computer Systems Science

& Engineering 2000; 15(5): 315-326. Special Issue on Flexible Workflow Technology Driving the Networked Economy.

[24] Rodden T. Populating the Application: A Model of Awareness for Cooperative Applications. Proceedings ACM Conference on Computer-Supported Cooperative Work (CSCW), 1996. p. 87-96.

[25] Roussev V, Dewan P, Jain V. Composable Collaboration Infrastructures Based on Programming Patterns. Proceedings ACM Conference on Computer Supported Cooperative Work (CSCW), 2000. p. 117-126.

[26] Schmidt K, Simone C. Coordination mechanisms: Towards a conceptual foundation of CSCW systems design. Computer Supported Cooperative Work: The Journal of Collaborative Computing 1996; 5(2-3): 155-200.

[27] Singhal S, Zyda M. Networked Virtual Environments: Design and Implementation, Addison-Wesley, 1999.

[28] Szyperski C. Component Software: Beyond Object-Oriented Programming, Addison-Wesley, 1998.

[29] Zhou Y et al. Fuzzy-Timing Petri Net Modeling and Simulation of a Networked Virtual Environment – NICE. IEICE Transactions on Fundamentals of Electronics, Communication and Computer Science, PART A, 2000; E83-A(11): 2166-2176.

A COMPONENT-BASED INFRASTRUCTURE FOR THE COORDINATION OF COLLABORATIVE ACTIVITIES IN VIRTUAL ENVIRONMENTS

Alberto B. Raposo1, **Christian M. Adriano**1,
Adailton J. A. da Cruz1,2, **Léo P. Magalhães**1

1 Department of Computer Engineering and Industrial Automation
School of Electrical and Computer Engineering – State University of Campinas
CP 6101 – 13083-970 – Campinas, SP, Brazil
2 Universitary Center of Dourados – Federal University of Mato Grosso do Sul
CP 322 – 79804-970 – Dourados, MS, Brazil
{alberto, medeiros, ajcruz, leopini}@dca.fee.unicamp.br

Abstract. *In this paper we introduce an approach based on software components to coordinate the behavior of virtual environments from the collaboration point-ofview, in which human or virtual participants interact by means of interdependent tasks. The coordination components communicate with tasks, controlling their execution and ensuring that interdependencies are not violated. By the use of coordination components, we not only separate the coordination model from the implementation of virtual environments, but also open the possibility to create a library of reusable components. The library implements the last part of a threesome coordination infrastructure for virtual environments, which is based on an abstract model, its mathematical counterpart, and its executable elements (the components).*

Keywords: Collaborative Virtual Environments, Coordination, Software Components, Modeling of Behavior, Computer Supported Cooperative Work.

1. Introduction

A networked virtual environment is a simulation of a real or imaginary world where multiple users may interact with each other in real time, share information, and manipulate objects in the shared environment [Singhal 99]. Due to these characteristics, virtual environments (VEs) appear as potential tools for the support of collaborative activities. Nevertheless, the development of VEs has been dominated by leisure activities, enabling basically navigation through virtual scenarios and communication with remote users. This kind of activity is what we call "loosely coupled collaborative activity" and is well coordinated by the social protocol, characterized by the absence of any explicit coordination mechanism, trusting the participants' abilities to mediate interactions.

On the other hand, "tightly coupled collaborative activities" require sophisticated coordination mechanisms in order to be efficiently realized in VEs. This is a kind of activity whose tasks depend on each other to start, to be performed, and/or to finish. Examples of tightly coupled collaborative activities include collaborative authoring, workflow procedures, computer games, among others. A core concern in collaborative Ves is the risk that actors get involved in conflicting or repetitive tasks. The coordination SVR

2001 - 4th SBC Symposium on Virtual Reality, p. 127 - 138. Florianópolis, Brazil. SBC, 2001. needed in this context, is defined as “the act of managing interdependencies between activities performed to achieve a goal” [Malone 90] and is managed by coordination mechanisms [Schmidt 96], which are software devices that interact with the application to control the behavior of the VE.

We argue here that an effective solution for the coordination issue resides in an infrastructure-like initiative. Such initiative consists of thinking the solution as free of application domain details. Instead, the infrastructure should be viewed as three abstract aspects inherent to coordination dependent domains, namely a generic coordination model and its mathematical and executable counterparts. The reason for this abstraction resides on two characteristics of coordination. First, many coordination features are generic and apply to domains out-side VR research, such as time sharing, process scheduling, locking policies, race condition analysis, deadlock prediction, etc. Due to all these computer pervasive facts, any solution for the coordination issue has necessarily a considerable generic dimension. Therefore, an infrastructure-like solution is convenient, since its essence is to provide some set of general functionality. As second point, we aimed at a solution to be as independent as possible of the specific VE semantics. Such feature would enable decoupling tradeoffs between a coordination apparatus and the whole VE.

The coordination infrastructure we propose in this paper is based on the following tripod: task-interdependency model, a mathematical formalism, and a component-based framework. The task-interdependency model abstracts the aspects necessary to coordination reasoning and actuating. The formalism validates and implements the mathematical aspects of the model, providing tools for prediction and optimization. The component-based framework implements the executable aspects of the model.

The paper is organized as follows. Section 2 describes the coordination model and briefly depicts the mathematical formalism. Section 3 explains the framework of coordination control. An implementation using the presented infrastructure was crafted in a form of videogame prototype (Section 4). The last section presents the conclusions.

2. Coordination Model

The required model must represent the VE in a manner that a coordination control could reason on it and actuate effectively. The coordination model presented here stems from two previous results accomplished in earlier work [Raposo 00a].

The coordination model, which constitutes the first element of the infrastructure, defines a collaborative activity as an interdependent set of tasks realized by multiple actors to achieve a common goal. Also according to this model, tasks are the building blocks of activities and may be atomic or composed of subtasks. Tasks are connected to one another through interdependencies, and the management of such interdependencies is realized by coordination mechanisms. Interdependencies in the proposed model are divided into two types, *temporal* and *resource management* [Raposo 00a].

Temporal interdependencies establish the execution order of tasks. The set of temporal interdependencies is based on temporal relations defined by Allen [Allen 84]. According to him, there is a set of primitive and mutually exclusive relations that could be applied over time intervals (and not over time instants). This characteristic made these relations suited for task coordination purposes, because tasks are generally non-instantaneous operations.

This temporal logic, however, is defined in a context where it is essential to have properties such as the definition of a minimal set of basic relations, and the mutual exclusion among them. Temporal interdependencies between collaborative tasks, on the other hand, are inserted in a different context. For this reason, it was necessary to adapt Allen's basic relations, adding a couple of new relations and creating variations of those originally proposed. The main difference in the context of collaborative activities is that it is possible to relax some restrictions imposed by the original relations. The result of the adaptation for the context of collaborative activities is a set of 13 temporal interdependencies presented elsewhere [Raposo 00a]. To illustrate some of these interdependencies, consider two tasks T1 and T2 that occur, respectively, in intervals [t1i, t1f) and [t2i, t2f).

T1 equals T2 (t1i = t2i and t1f = t2f): This dependency establishes that two tasks must occur simultaneously.

T2 afterA T1 (t1f,n < t2i,n, □n > 0, where *n* means the nth execution of the task): T2 may only be executed if T1 has already finished. In this case, T2 may be executed only once after each execution of T1.

T2 afterB T1 (t1f,1 < t2i,n, □n > 0): Variation of the previous dependency, in which T2 may be executed several times after a single execution of T1.

T1 meets T2 (t1f = t2i): T2 must start immediately after the end of T1.

Resource management interdependencies are complementary to temporal ones and may be used in parallel to them. This kind of interdependency deals with the distribution of resources among the tasks. By "resource" we mean not only the agent that performs the task, but also any artifact needed to the execution of the task. Three basic resource management dependencies were defined:

Sharing: A limited number of resources must be shared among several tasks. It represents a situation that occurs, for example, when several users want to edit a document.

Simultaneity: A resource is available only if a certain number of tasks request it simultaneously. It represents, e.g., a machine that may only be used with two operators.

Volatility: Indicates whether, after the use, the resource is available again.

The second element of our threesome infrastructure is the mathematical formalization of coordination mechanisms that control the above interdependencies. The models of the coordination mechanisms are used for the analysis and simulation in order to verify the correctness and validate the efficiency of the collaborative environment before its implementation, which is an important step in the design of complex environments.

The coordination mechanisms have been modeled using three distinct approaches, conventional [Raposo 00a], high level [Raposo 00b], and fuzzy [Raposo 01] Petri Nets (PNs). The choice of PNs as modeling tool is justified because they are a well-established theory (there are numerous applications and techniques available) and can capture the main features of VEs, such as concurrency, synchronization of asynchronous processes and nondeterminism. Moreover, PNs accommodate models at different abstraction levels and are amenable both to simulation and formal verification.

The goal of the present work is to give the next step on the coordination infrastructure, defining a software component framework to implement those coordination mechanisms as

"black boxes" connecting the collaborative tasks. The coordination components are capable of receiving and generating events from/to collaborative tasks, controlling their execution. Thus, events related to tasks (e.g., the beginning of a task) may generate output events that affect the execution of interdependent tasks (e.g., blocking the execution of another task). The components have been implemented as a set of JavaBeans [JavaBeans 01], which is a framework consisting of a Java API that defines rules by which to implement software components (beans) and to have them integrated through an event notification scheme.

3. Coordination Components Framework

Software components [Szyperski 98] were idealized based on an analogy with electronic components, characterized by reusable modules with well-defined functions. Today, issues such as autonomy, interconnection and integration complement the component approach, in addition to reuse and modularization. The paradigm states that components are integrated in order to provide complete functions, but neither keep references to one another nor communicate directly–the communication is achieved by means of messages. The decoupling compromise satisfies a number of necessities, such as the exchange of a component for another and the immediate insertion of new components.

Therefore, the component model is a satisfactory solution for the design of Ves that follow the task/interdependency model for tightly coupled collaborative activities. Allied with the component paradigm is the framework design approach, which together perform the executable aspect of the proposed coordination infrastructure.

The component framework is presented under two documentation approaches, a pattern-based [Johnson 92] and hook methods and hotspots [Albrecht 97]. The former explains the framework design as problem/solution pairs. The latter establishes the effective bridges between requirements and design solutions. The framework will be described as follows: requirements met, internal flow of control, and integration and extension.

3.1 Requirements

We idealized the coordination level as a software layer loosely coupled to the scene (VE). The advantages of this approach are twofold. It isolates the scene from changes made exclusively in the coordination control architecture, and also provides a clear test bench to investigate coordination control issues. A number of requirements should be supported by the coordination control layer: (i) implementation separated from the scene; (ii) accommodation of changes in the scene, such as removal or inclusion of a task; (iii) possibility of changes in the coordination logic, which should be localized; (iv) mixing of different coordination logic within the same coordination control and (v) attachment of monitoring tools to gather performance related to the coordination control strategy (potential evaluation indices are throughput, deadlock risk, and load balance).

The architectural decoupling between components is the solution for requirements *i-iii*. The encapsulation implementation of complete services within each component is the solution for requirement *iv*. The event-oriented communication is the basis to attach external tools (requirement *v*).

3.2 Internal Flow of Control

The flow of control is basically an algorithm. Since a framework is a kind of incomplete application, it also encapsulates an algorithm. Here we depict this algorithm from a high level vision of components involved in the coordination between two tasks.

Figure 1 illustrates such high level vision. In the coordination level the figure shows three components, a coordinator and two tasks. The coordinator component implements the coordination mechanisms, both for temporal and resource management dependencies. The task component, instantiated for each task, is responsible for maintaining the task's schedule. The components of our architecture were designed to communicate by means of the sequence of events described in Table 1.

In the model of the coordinator component, a task has to wait until the temporal and resource management coordination mechanisms authorize its execution. A consequence of this fact is that the task may wait indefinitely if either of these conditions is not satisfied. In order to avoid such situations, the coordinator sends timeout signals, described in Table 2. The treatment of timeouts is left to the task components. This gives more flexibility to the architecture, because it does not risk the coordinator component reusability.

A possible consequence of timeouts is that the non-execution of an expected task may invalidate interdependent tasks previously executed. For this reason, the coordinator component sends timeout signals to all interdependent tasks, and not only to that which had its execution unauthorized.

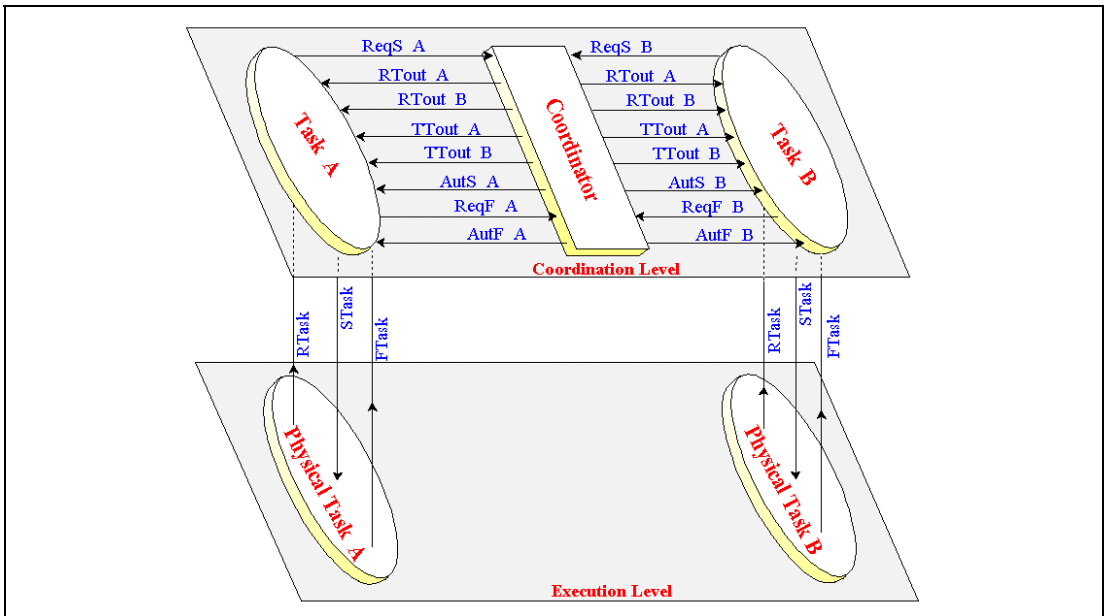


Figure 1: High-level vision of the coordination component acting between two tasks.

Up to this point we have centered the discussion in the coordination level, where only the logical execution of the tasks is considered. Figure 1 also illustrates the execution level that represents the "physical" execution of the tasks in the VEs. The connection between both levels is left to the task components, reducing the responsibilities of the

coordinator components, what contributes for their reuse. The physical tasks communicate with their respective task components by means of the events described in Table 3.

An advantage of this two-level approach is the modularization of the architecture. The coordination model of the VE may be developed independently of its implementation and vice versa. The only compromise between both levels is the communication by means of the three signals described in Table 3.

EVENT	DESCRIPTION
REQUEST START (ReqS)	When a task is demanded, for example due to a user action, it must first contact the coordinator to request an authorization for execution. At this time, the coordinator checks if there is any resource management interdependency and, if so, it consults the resource management mechanism to verify if the resource is available (if not, it waits until the resource becomes available). Once the resource is available or in the case of having no resource dependency, the coordinator checks if there is any temporal interdependency. If so, it consults the temporal coordination mechanism to verify if the conditions to the task's beginning are satisfied (if not, it waits until these conditions are satisfied). Once all conditions are satisfied, the signal AutS is sent to the task component.
AUTHORIZE START (AutS)	This signal is the authorization given by the coordinator to enable the beginning of a task's execution.
REQUEST FINISH (ReqF)	Once the task wants to finish its execution, it sends this signal to the coordinator, which verifies if the temporal interdependency (whether it exists) enables the end of the task. If so, it sends the signal AutF to the task and, in the case of having a resource management dependency, releases the assigned resource. Otherwise, the coordinator waits until the temporal coordination mechanism authorizes the task's finish.
AUTHORIZE FINISH (AutF)	This signal indicates to the task that it may finish.

Table 1: Events between coordination and task components.

EVENT	DESCRIPTION
RESOURCE TIMEOUT (RTout)	If the resource is not assigned to the task after a certain waiting time, the coordinator sends this signal.
TEMPORAL TIMEOUT (TTout)	If another task does not offer the conditions to the beginning of the task that requested it, the coordinator sends this signal after a certain waiting time. In this case, if the resource has already been assigned to the task, the coordinator also releases it.

Table 2: Timeout signals.

EVENT	DESCRIPTION
REQUEST TASK (RTask)	This signal is sent from the physical task to the respective component when an event in the VE (user action, solicitation of another task, elapsed time, etc.) asks for its beginning. The task component forwards this information to the coordinator in the coordination level via ReqS signal.
START TASK (STask)	When the coordinator authorizes the beginning of a task (AutS), the task component forwards this authorization to the physical task via this signal. At this moment the task really starts its execution in the VE.
FINISH TASK (FTask)	The end of the task's execution in the VE generates this signal that is sent to the respective component in the coordination level. This information is forwarded to the coordinator via ReqF signal. It is important to clarify that the task's logical end (i.e., from the coordination point-of-view) occurs when the task component receives the AutF signal. The logical end of a task guarantees that temporal interdependencies are not violated and may be delayed in relation to the physical end in specific situations.

Table 3: Events between the physical task and the task component.

Concerning the coordination component internals, it encapsulates two PN simulators, one for the temporal and another for the resource management coordination mechanism. Each of these simulators interacts with their associated events, received or sent by the component. Events arriving at the coordinator alter the state of the simulator, while certain states of the simulator generate output events. Figure 2 depicts the coordinator functioning.

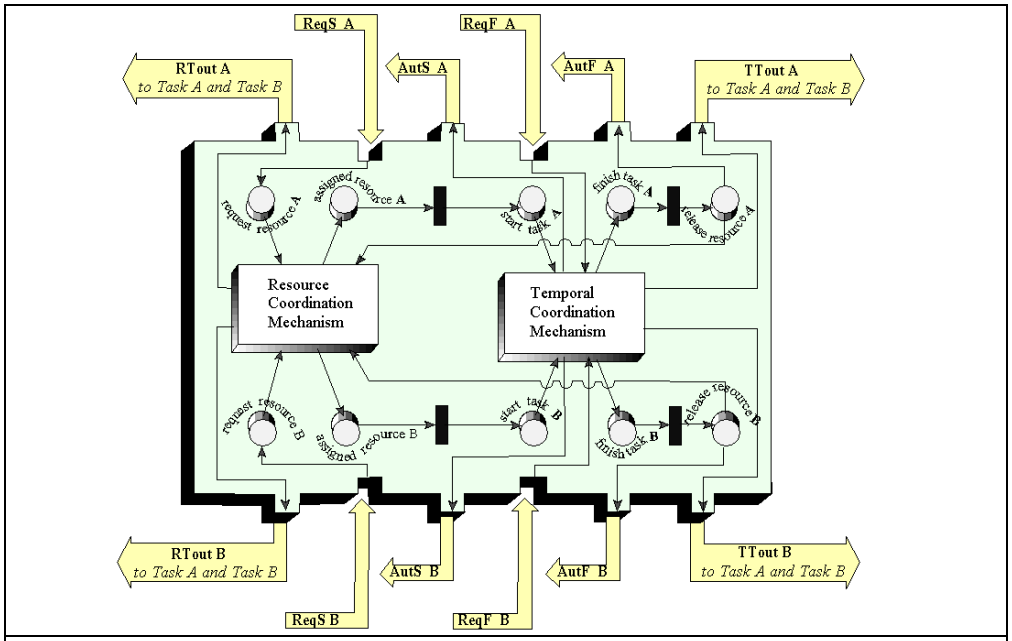


Figure 2: The coordinator functioning.

When the coordinator receives a ReqS signal, it puts a token in the respective request_resource place, starting the resource management mechanism. When the resource is available, the resource management mechanism puts a token in the assigned_resource place, which sends it to start_task. The start_task place starts the temporal coordination mechanism, which sends the respective AutS signal when the task may begin. When the physical task is finished, the temporal mechanism receives the ReqF signal. If the logical end of the task is authorized, a token is sent to finish_task, and then to release_resource, which indicates to the resource management mechanism that the resource is free. Finally, the coordinator sends the AutF signal, indicating the logical end of the task. The PN models for several temporal and resource management mechanisms have been shown elsewhere [Raposo 00a]. Although belonging to the same component, both internal coordination mechanisms have independent behaviors, running different PN simulators.

3.3 Integration and Extension

Although the only compromise between the coordination and execution levels is the communication by means of the signals discussed above, the integration issue is not an easy task, specially due to the tight relation between scene semantics and its coordination. The idea is that the decoupled integration resides essentially on communication patterns between the scene and coordination components. A communication pattern imposes different kinds of externalization of scene internal elements (i.e., actors, resources, tasks, etc.). In the following, we discuss three externalization possibilities (Figure 3).

The first possibility is to implement the VE under the software component paradigm (i.e., actors, tasks, and resources implemented as components), which would lead to a straightforward integration with the coordination control. Each task component in the coordination level would be connected to a set of components in the scene. In this approach connections are loosely coupled because TaskBean (Figure 3a) does not need to import any scene library/package. This fully component-based integration approach is the ideal one from the software architecture point-of-view, but it requires a component-based VE.

A second alternative would be to encapsulate the whole scene in a single IntegratorBean, which would be basically responsible to forward coordination commands to the appropriate scene elements and to return scene events to respective task components at the coordination level (Figure 3b). This single point integration approach solution has the advantage of not imposing a reorganization of scene elements, enabling different scene technological implementations. Its drawback is a less decoupled and robust solution, since even a slight change in the scene would impact the IntegratorBean.

A third approach is to have each task component accessing their respective scene elements (Figure 3c). The advantage of this direct integration approach is that there is not a unique integration point and, therefore, no need to translate JavaBeans events to method calls in scene elements. Moreover, there is no need to restrict or adapt the scene to the coordination components. Scene elements may even be implemented as simple structured and not object-oriented code. The drawbacks, however, are numerous. The sense of layer is wicked, because coordination components need to import scene libraries. Furthermore, even a single change in the scene could impact the coordination components in a less uniform and localized manner than in previous integration approaches.

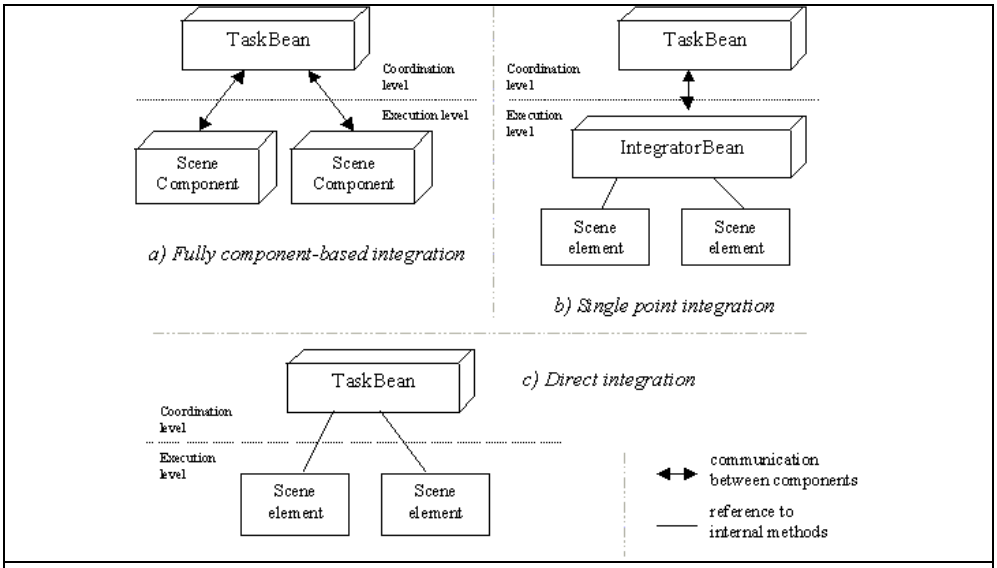


Figure 3: Approaches for the integration of scene-coordination components.

Another framework design concern is the extension issue. There are three concepts needed here, hotspot subsystem and its two methods, hook and template. Hotspots represent a framework's adaptable aspects [Pree 99] and must be extended by the application in order to be active (though default implementations must also be provided). In concrete means, the hotspot is a set of concrete and abstract classes. The hook method is defined in one of the abstract classes, while its implementation is in application code. Furthermore, the hook method is called by a framework private method, namely a template method.

Two hotspot subsystems with default extensions were designed and are shown in Figure 4. The first hotspot (Figure 4a) consists of the TaskComponent, which implements the integration choice between coordination and execution layers. The TaskBean component, presented in the previous integration discussion, appears here as the default implementation for the TaskComponent. The second hotspot subsystem (Figure 4b) consists of class CoordinationComponent. This hotspot effectively enables the logic to be adapted, which is done by inheriting the CoordinationComponent. The default implementation uses a PN logic. Additionally, the classes in the diagram define protected (#) properties needed by the inter-component message exchange and declare the public (+) hook methods that are called by private (-) template methods. These last ones are not shown here.

4. Example: A Multiuser Videogame

In this section it is presented a case study of a VE where a user interacts with an autonomous agent that represents a second user. The example implements a kind of videogame based on the second "task" (activity) of Heracles, from the Greek mythology. According to the legend, Heracles had to kill the Hydra of Lerna, a monster with nine heads that are regenerated after being severed. In order to achieve his goal, Heracles needs the collaboration of his nephew Iolaus, who cauterizes the monster's wounds after Heracles

cuts off each head. However, the last head may not be severed by any weapon. The solution was to bury the monster in a deep hole and cover it with a huge stone.

Figure 5 illustrates an abstract PN-like model of the videogame (open rectangles indicate interdependent tasks). There are two identical nets, one representing the user's and the other representing the agent's sequence of tasks. Each net has two alternative paths, indicating that each "actor" (user or agent) may assume either role (Heracles or Iolaus). The upper part of the nets represents Heracles' sequence of tasks. He must get the sword, sever eight of the Hydra's heads, throw the beast into the hole and cover it with a stone. The lower part of the nets represents Iolaus' sequence of tasks. He must get the torch, cauterize the wounds after Heracles has severed the heads and dig the hole. The boxes with interdependency names are substituted by the respective coordination mechanism models for simulation and analysis purposes. In the implementation, these boxes represent the internal mechanisms of the coordination components to be used.

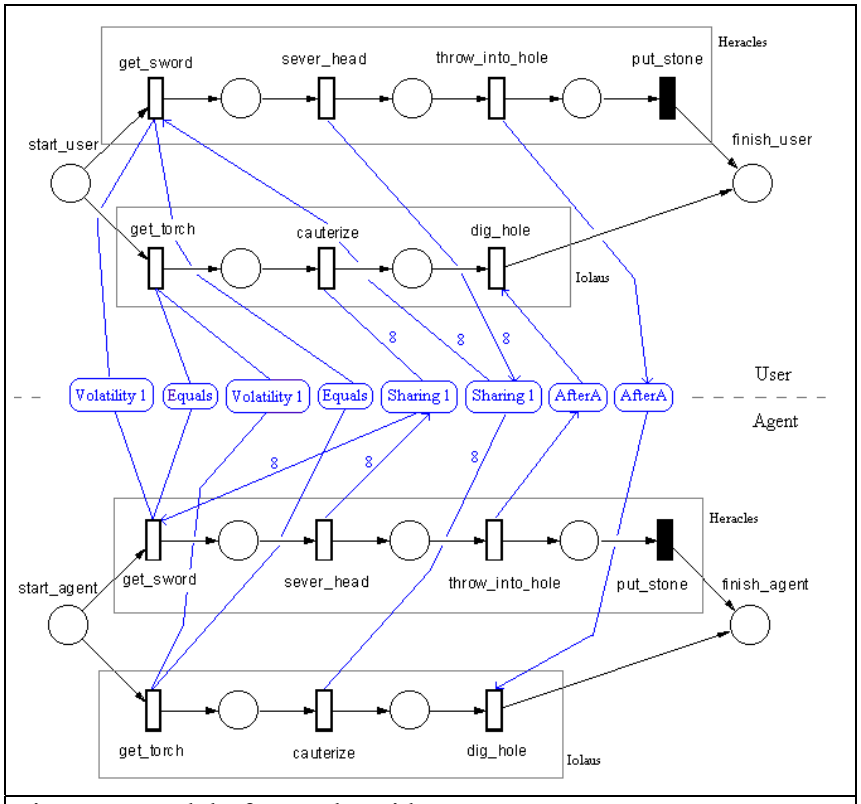


Figure 5: Model of Heracles videogame

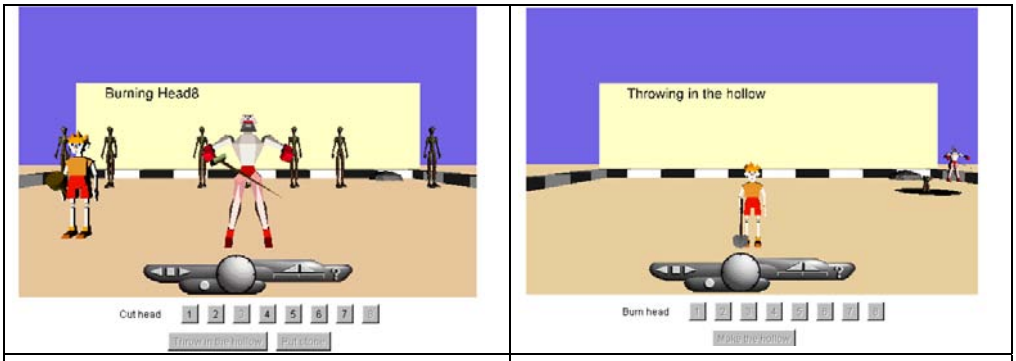
The definition of which actor will assume which role is given by the interdependencies volatility 1 between tasks get_sword and get_torch. Since there are only one sword and one torch available, the weapon's choice determines that each actor will assume a different role. There is also an equals interdependency between get_sword and get_torch of different actors, forcing the agent to choose the other weapon when the user chooses his/her weapon.

The interdependency sharing 1 among the tasks get_sword, sever_head and cauterize is the "core" of the game. When Heracles gets the sword, he is also assigned eight

resources that may be thought as “abstract authorizations” to cut Hydra’s heads. After he has severed each head, a resource is released, indicating to Iolaus that he may cauterize that wound. If the head wound is not cauterized within a certain period after it has occurred, the timeout signal sent by the coordination mechanism causes the return of the task to its initial state and also reassigns the resource to Heracles, indicating that he must sever that head again (the head is regenerated). There is also an interdependency afterA, indicating that Heracles may only throw the monster in the hole if Iolaus already has dug it.

The videogame was implemented using the blaxxun Contact [Blaxxun 00], a client for multimedia communication that provides resources for VRML (Virtual Reality Modeling Language) visualization, chats, message boards, avatars, etc. In this implementation we used the VRML visualization and the avatars with pre-defined movements.

The interaction with the user occurs by means of buttons defined in a Java applet that interacts with the VRML world via EAI (External Authoring Interface), an interface that enables external programs to interact with objects of a VRML scene. By clicking on the applet’s buttons, the user orders the execution of a task in the virtual world. Therefore, the coordination mechanisms act on the interface’s buttons, enabling or disabling them if their respective tasks are enabled or not. In order to give more dynamism for the game, the agent has an aleatory behavior, taking a variable time to start the execution of the tasks imputed to it. For example, when the user assumes the role of Heracles, the agent (Iolaus) may not cauterize a head wound before it is regenerated. Figure 6 shows some frames of the videogame (the nine heads of Hydra are represented by nine monsters). Frame a shows Heracles’ interface and frame b shows Iolaus’ interface.



a) **b)**

Figure 6: Frames of Heracles videogame.

5. Conclusion

The main goal of this work was to create facilities for the design and implementation of VEs. Due to the threefold infrastructure, we established the basis to intertwine the abstract coordination model with the mathematical formalism and the executable code. Concerning the executable part, the choice for software components and for structuring them in a framework incurred in two main benefits, the concrete realization of the coordination layer as idealized, and the flexibility to change the coordination logic that was initially crafted with a fixed mathematical formalism (PNs).

Our coordination approach follows a multi-abstraction levels hierarchy, which has the advantage of isolating the parts of coordination design. For instance, if the VE designer wants to implement a videogame such as the one presented in Section 4, it is not necessary to consider the mathematical model of the system. Designers may simply use pre-defined coordination components, which encapsulate the formal logic of the coordination mechanisms. On the other hand, in initial phases of the design, only the mathematical model may be used, not considering implementation details.

Currently, collaborative VEs have been mainly developed based on a trial and error approach. We believe that the presented infrastructure is a step towards offering a systematic approach for the development of this kind of environment. One of the few work results that propose another systematic approach for that is [Kim 98], which proposed the use of CASE tools for VE's development.

As future research, we plan to use the infrastructure not only for the implementation of more complex VEs but also for the implementation of other kinds of collaborative applications, making the necessary adjustments.

Acknowledgments. A. Raposo is sponsored by FAPESP (00/10247-3), and A. Cruz, by CAPES/PICD. Thanks also to Tecgraf / PUC-Rio for the expressive support.

References

[Albrecht 97] H. Albrecht. Systematic Framework Design, *Communications of the ACM*, 40(10): 48-51, Oct 1997.

[Allen 84] J. F. Allen. Towards a General Theory of Action and Time. *Artificial Intelligence*, 23: 123-154, 1984.

[Blaxxun 00] blaxxun interactive. *blaxxun Contact 4.4*. <http://www.blaxxun.com/products/contact>, Aug 2000.

[JavaBeans 01] Sun Microsystems. *JavaBeans*. <http://java.sun.com/products/javabeans>, May 2001.

[Johnson 92] R. Johnson. Documenting Frameworks with Patterns, *Proc. of OPSLA'92,* pp. 63-76, 1992.

[Kim 98] G. J. Kim et al. Software Engineering of Virtual Worlds. *Proc. of VRST'98*, pp. 131-138, 1998.

[Malone 90] T. W. Malone and K. Crowston. What is Coordination Theory and How Can It *Help* Design Cooperative Work Systems? *Proc. of CSCW'90*, pp. 357-370, 1990.

[Pree 99] W. Pree. Hot-Spot-Driven Development. In *Building Application Frameworks, Object-Oriented Foundations of Framework Design*, M. E. Fayad, D. Schmidt, and R. Johnson (Ed.). Wiley, 1999.

[Raposo 00a] A. B. Raposo, L. P. Magalhães and I. L. M. Ricarte. Petri Nets Based Coordination Mechanisms for Multi-Workflow Environments. *Int. J. Computer Systems Science & Engineering*, 15(5): 315-326. Sep 2000.

[Raposo 00b] A. B. Raposo, L. P. Magalhães and I. L. M. Ricarte. Coordinating Activities in Collaborative Environments: A High Level Petri Nets Based Approach. *Proc. of SCI'2000, Vol. I - Information Systems*, pp. 195-200, 2000.

[Raposo 01] A. B. Raposo, A. L. V. Coelho, L. P. Magalhães and I. L. M. Ricarte. Using Fuzzy Petri Nets to Coordinate Collaborative Activities. *Joint 9th IFSA World Congress and 20th NAFIPS Int. Conf.*, pp. 1494-1499, 2001.

[Schmidt 96] K. Schmidt and C. Simone. Coordination mechanisms: Towards a conceptual foundation of CSCW systems design. CSCW, 5(2-3): 155-200, 1996.

[Singhal 99] S. Singhal and M. Zyda. *Networked Virtual Environments: Design and Implementation*, Addison-Wesley, 1999.

[Szyperski 98] C. Szyperski. *Component Software: Beyond Object-Oriented Programming*. Addison-Wesley, 1998.